НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення веб-технологій та мобільних пристроїв»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Інструментальні засоби мобільного застосунку для організації контролю відвідувань студентів»

Виконав:
студент IV курсу, групи ТІ-62
Коваленко Дмитро Русланович _____

Керівник:
к.т.н., доцент
Гагарін Олександр Олександрович _____

Рецензент:
к.т.н., ст. викладач
Сірий Олександр Анатолійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення веб-технологій та мобільних пристроїв

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
_____Коваленку Дмитру Руслановичу_____
(прізвище, ім’я, по батькові)

1.Тема роботи Інструментальні засоби мобільного застосунку для організації контролю відвідувань студентів
керівник роботи Гагарін Олександр Олександрович, к.т.н., доцент
затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Мова програмування Dart, Набір інструментів від Google для створення користувацького інтерфейсу Flutter, Сервіс Firebase.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Провести дослідження сучасних цифрових методів оцінювання та організації контролю відвідувань студентів, Провести порівняльний аналіз існуючих аналогів, в якості мобільних застосунків, Визначити головні функціональні можливості застосунку на базі поставлених цілей та виявлених недоліків, знайдених в аналогах, Створити концепт-діаграму користувацького досвіду (UX) та визначити дизайн користувацького інтерфейсу (UI), Спроектувати модель класів, Обрати технології для реалізації мобільного застосунку, систему Firebase, в якості back-end частини та інтегрувати її до мобільного додатку для отримання доступу до хмарного сховища, Розробити необхідні віджети застосунку та об’єднати їх, Розробити функціональні можливості мобільного застосунку, Імплементувати запланований дизайн застосунку, Об’єднати всі складові програмного забезпечення, в якості фінального програмного продукту, Протестувати мобільний застосунок та виправити можливі помилки і покращити проблемні функціональні можливості.
5. Перелік ілюстративного матеріалу Електронний журнал, Проблема, Тенденція

сучасного світу, Рішення, Поставлені задачі, Схема роботи мобільного застосунку, Діаграма класів мобільного застосунку, Структура бази даних, Діаграма прецедентів мобільного застосунку, Сценарії роботи частини застосунку, що відповідає за авторизацію користувача, Вкладена навігаційна система, Сценарій створення нової групи студентів та створення курсу, Сценарій створення сесії певного лекційного курсу та проведення контролю відвідування студентів, Сценарій проведення оцінювання студентів практичного курсу, Сценарій керування даними окремого студента, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”11” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	15.10.2019	
2.	Вивчення та аналіз задачі	10.11.2019	
3.	Розробка архітектури та загальної структури системи	12.12.2019	
4.	Розробка структур окремих підсистем	10.03.2020	
5.	Програмна реалізація системи	26.04.2020	
6.	Оформлення пояснювальної записки	18.05.2020	
7.	Захист програмного продукту	09.06.2020	
8.	Передзахист	09.06.2020	
9.	Захист	16.06.2020	

Студент _____ _____Коваленко Д. Р._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Гагарін_О.О.__
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Обсяг звіту становить 64 сторінку, містить 46 рисунків, 3 додатки. Загалом опрацьовано 10 джерел.

Метою роботи було дослідження методів розробки програмного забезпечення для мобільних пристроїв, а саме смартфонів, патернів UI та UX, і створення програми, що буде відповідати кращим стандартам даної індустрії. Для реалізації було обрано створення мобільного застосунку для оцінювання робіт студентів та організації контролю їх відвідувань .

Проведено дослідження сучасних цифрових методів оцінювання та організації контролю відвідувань студентів. Проведено порівняльний аналіз існуючих аналогів. Спроектовано програмний продукт. Обрано технології для реалізації мобільного застосунку. Розроблено всі складові програмного забезпечення, в якості фінального програмного продукту. Протестовано мобільний застосунок та виправлено помилки.

Ключові слова: мобільний застосунок, користувацький інтерфейс, користувацький досвід, порівняльний аналіз, аналоги, система Firebase, віджети, дизайн, Flutter, Dart.

ABSTRACT

The scope of the report is 64 pages, which contains 46 figures, 3 appendices. A total of 10 sources were processed.

The aim of the work was to study the methods of software development for mobile devices, namely smartphones, UI and UX patterns, and to create a program that will meet the best standards of the industry. For implementation, it was chosen to create a mobile application for evaluating the work of students and organizing the control of their visits.

A study of modern digital methods of assessment and control of student attendance. A comparative analysis of existing analogues as mobile applications. The software product is designed. Selected technologies for the implementation of mobile applications. All components of the software have been developed as a final software product. Mobile application tested and bugs fixed.

Keywords: mobile application, user interface, user experience, comparative analysis, analogues, Firebase system, widgets, design, Flutter, Dart.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 8
ВСТУП 9
1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ КОНТРОЛЮВІДВІДУВАНЬ СТУДЕНТІВ 11
2 АНАЛІЗ UI/UX РІШЕНЬ ТА ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ АНАЛОГІВ ЗАСТОСУНКУ 14
2.1 Важливість UI/UX дизайну 14
2.2 Застосунок «Class Register» 17
2.3 Застосунок «Easy Attendance» 22
2.4 Застосунок «Attendance taker» 27
2.5 Висновки до розділу 29
3 НАБІР ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ UI "FLUTTER" ТА BAAS FIREBASE 31
3.1 Flutter 31
3.2 Dart 34
3.3 Firebase 37
3.4 Висновки до розділу 41
4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 42
4.1 Структура програми 42
4.2 Опис розроблених алгоритмів 44
4.3 Керівництво користувача ..... **Ошибка! Закладка не определена.**
4.4 Висновки до розділу 61
ВИСНОВКИ 63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 64
ДОДАТОК А 65

ДОДАТОК Б 67
ДОДАТОК В 85
ДОДАТОК Г 96

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI	–	User Interface
UX	–	User Experience
BAAS	–	Backend As A Service

ВСТУП

На сьогодні викладачі в університетах ще досі використовують старі методи для контролю відвідувань та оцінювання студентів. Великий обсяг паперів зі списками студентів для різних груп і предметів не є достатньо надійним та зручним способом організації інформації про навчальний процес.

Варто зазначити, що ненадійність, часом, може привести до неприємних наслідків, таких як втрата важливих даних, від яких залежить оцінювання студентів. Також, не менш проблематичним буває й сам процес виставлення оцінок і контроль відвідування студентами лекцій.

Серед тенденцій сучасного світу усе більших масштабів набирає використання мобільних застосунків для спрощення доступу до необхідної інформації. Спочатку мобільні програми пропонувались як для моніторингу, так і для управління загальними інформаційними потоками, включаючи електронну пошту, календар, контакти, інформацію про фондову біржу та погоду. Однак попит та доступність інструментів для розробників призвели до швидкого розповсюдження застосунків для інших категорій електронних пристроїв, які працюють за допомогою настільних застосунків.

Використання мобільних застосунків серед користувачів мобільних пристроїв стає все більш популярним. Згідно з дослідженням AppAnnie в 2017 році, кількість завантажень застосунків збільшилася на 60%, споживчі витрати зросли більш ніж удвічі, а час, витрачений на застосунок кожним користувачем, становив близько 43 днів на рік. [1]

У 2014 році державні регуляторні служби України почали створювати та регулювати мобільні застосунки, зокрема ті, що стосуються медичної галузі. Деякі компанії навіть почали пропонувати мобільні програми як альтернативний метод надання інформації, на відміну від офіційних сайтів. Наприклад, у 2017 році понад

60% трафіку на веб-сайти українських мікрофінансових компаній надійшло з мобільних пристроїв. [2]

Мобільні застосунки відіграють все більш важливу роль у галузі охорони здоров'я та освіти, і при правильному проектуванні та інтеграції вони можуть забезпечити багато переваг. AppAnie також прогнозує зростання на 270% ринку мобільних додатків до 2020 року. [1]

Для вирішення проблеми не достатньо надійного та не зручного способу організації інформації про навчальний процес були створені такі аналоги: “Class Register”, “Easy Attendance”, “Attendance Taker”. Проте, проблема залишається не вирішеною, адже аналоги мають незручний інтерфейс і складний досвід користувача (UX) або недостатні функціональні можливості.

Опираючись на вказані причини, вбачається доцільним розробка мобільного застосунку, який вирішує дану проблему і є легким у використанні.

Для розробки програмного застосунку було обрано мову програмування Dart та набір інструментів для створення користувацького інтерфейсу Flutter. Розробка відбувалась у середовищі програмування Microsoft Visual Studio Code. В якості back-end частини був використаний сервіс Firebase, а саме Authentication Service для створення розширених можливостей авторизації та Cloud Firestore в якості системи управління базами даних.

Перший розділ пояснювальної записки містить постановку задачі. У другому розділі описується проведений аналіз UI / UX рішень та функіцональних можливостей аналогів застосунку. У третьому розділі продемонстрована структура програмного забезпечення та керівництво користувача до створеного мобільного застосунку.

Таким чином, розроблені інструментальні засоби в якості мобільного застосунку можуть бути використані в усіх можливих навчальних закладах, де потрібно проводити контроль відвідування та оцінювання практичних робіт студентів.

1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ КОНТРОЛЮ ВІДВІДУВАНЬ СТУДЕНТІВ

Метою даної дипломної роботи є створення мобільного застосунку для оцінювання та організації контролю відвідувань студентів з інтуїтивним UI та легким UX і збереженням даних в хмарному сховищі.

При розробленні відповідного забезпечення потрібно розв’язати наступні завдання:

1. Провести дослідження сучасних цифрових методів оцінювання та організації контролю відвідувань студентів.
2. Провести порівняльний аналіз існуючих аналогів, в якості мобільних застосунків.
3. Визначити головні функціональні можливості застосунку на базі поставлених цілей та виявлених недоліків, знайдених в аналогах.
4. Створити концепт-діаграму користувацького досвіду (UX) та визначити дизайн користувацького інтерфейсу (UI).
5. Спроектувати модель класів.
6. Обрати технології для реалізації мобільного застосунку.
7. Налаштувати систему Firebase, в якості back-end частини та інтегрувати її до мобільного додатку для отримання доступу до хмарного сховища.
8. Розробити необхідні віджети застосунку та об’єднати їх.
9. Розробити функціональні можливості мобільного застосунку.
10. Імплементувати запланований дизайн застосунку.
11. Об’єднати всі складові програмного забезпечення, в якості фінального програмного продукту.

12. Протестувати мобільний застосунок та виправити можливі помилки і покращити проблемні функціональні можливості.

Вимоги до розроблюваного програмного засобу:

1. Інструментальні засоби мають зберігати дані в хмарному сховищі, і користувач має мати змогу отримати доступ до них на будь-якому іншому мобільному пристрої, де є застосунок, пройшовши аутентифікацію.
2. Інструментальні засоби мають складатися з чотирьох підсистем:
   - Система аутентифікації.
   - Система створення груп та предметів.
   - Система контролю відвідувань студентів на лекціях.
   - Система оцінювання студентів на практиках.
3. Вхідними даними системи аутентифікації мають бути електронна пошта та пароль або існуючий акаунт Google.
4. Вихідними даними системи аутентифікації мають бути інформація привязана до даного акаунту та дані для подальшої автоматичної аутентифікації.
5. Вхідними даними системи створення груп та предметів мають бути імена студентів, назви груп, назви предметів та їх класифікація.
6. Вихідними даними системи створення груп та предметів мають бути сутності студентів, груп та предметів.
7. Вхідними даними системи контролю відвідувань та оцінювання студентів на лекціях і практиках мають бути назви та дати створення уроку, мітка присутності на лекції та оцінка на практиці.
8. Вихідними даними системи контролю відвідувань та оцінювання студентів на лекціях і практиках мають бути збережені вхідні дані, відсоток відвідувань та їх кількість на лекціях або середнє арифметичне оцінок та їх сума на практиках для кожного студента.
9. Наявна можливість атворизації та регистрації.
10. Наявна можливість авторизації за допомогю акаунту Google.

11. Наявна можливість відновлення паролю до існуючого акаунту.
12. Наявна можливість входу до системи без авторизації та підключення до інтернету, якщо користувач вже входив до свого акаунту і не виходив з нього.
13. Наявна можливість автоматичного збереження всіх даних до хмарного сховища. Якщо пристрій не має підключення до інтернету, дані мають записуватись на локальне сховище і завантажуватись після відновлення підключення.
14. Наявна можливість створення груп зі студентами, предметів з класифікацією лекції та практики.
15. Наявна можливість зміни назви групи та імен окремих студентів, а також видалення студентів зі списку групи.
16. Наявна можливість видалення предметних практик або лекцій.
17. Наявна можливість автоматичної назви та нумерації уроків лекцій та практик кожного предмета.
18. Наявна можливість зміни назви або видалення уроку лекції або практики.
19. Наявна можливість автоматичної прив’язки дати до уроку при його створенні.
20. Наявна можливість зміни імені або видалення студента в межах одного предмету.
21. Наявна можливість переглянути сторінку окремого студента на котрій зібрані дані, щодо його відвідування та отриманих оцінок, а також відсоток відвідувань та їх кількість на лекціях або середнє арифметичне оцінок та їх сума на практиках.
22. Наявна можливість на сторінці студента змінювати дані, щодо його відвідувань та отримані оцінки.

2 АНАЛІЗ UI/UX РІШЕНЬ ТА ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ АНАЛОГІВ ЗАСТОСУНКУ

2.1 Важливість UI/UX дизайну

Для комерційного успіху будь-якого мобільного застосунку важливу роль займає його UI/UX дизайн. Адже, велика кількість функціональних можливостей застосунку завжди обмежена розміром екрану мобільного пристрою та «зоною великого пальця» на ньому.

Щоб зрозуміти роль UI/UX дизайну в розробці мобільних застосунків, спочатку потрібно визначити, що собою являє і які має цілі UI та UX дизайн окремо. Потім варто зрозуміти, чому їх поєднання є важливим, як вони впливають один на одного, та які правила є основними для їх створення.

Дизайн застосунку є об’єднанням функціональних можливостей, форми, властивостей та естетичних проявів.

UX (досвід користувача) дизайн відповідальний саме за представлення властивостей та функціональних можливостей користувачу в такому вигляді, щоб створений продукт був приємним у взаємодії та простим у використанні. Тож мета UX дизайну максимально просто привести користувача до його фінальної цілі та забезпечити вирішення проблеми.

UI (інтерфейс користувача) дизайн відповідальний за візуальне представлення застосунку, тобто вибір кольору, читабильність тексту, чи зручно користувачу натискати пальцем на кнопки.

Як результат, задача UI/UX дизайну – вплинути на правильний вибір користувача через створений інтерфейс. Створення всіх умов для того, щоб користувач дотримувався певного алгоритму дій при роботі з застосунком на інтуїтивному рівні.

Щоб почати створення UX, важливо проаналізувати існуючі аналоги, цільових користувачів та створити певну бізнес-модель.

Порівняльний аналіз аналогів майбутнього мобільного застосунку дозволяє:

1. Виявити слабкі та сильні сторони оцінюваних об’єктів.
2. Розширити розуміння цільової аудиторії та предметної області
3. Створити власне бачення проекту та згенерувати власні ідеї

Варто зазначити, що користь від вивчення цільової аудиторії є великою і надає такі переваги при проектуванні застосунку:

1. Визначені основні та другорядні цілі користувачів.
2. Визначена мотивація користувачів.
3. Виявлені типові проблеми користувачів.
4. Створений пріоритет потреб користувача.
5. Сформований портрет цільового користувача.

UX – дизайн, що оцінює всі аспекти взаємодії користувача з продуктом. Він сильно пов’язаний з розумінням поведінки користувача, його цілями, потребами, мотивацією та контекстом, в якому використовується продукт.

UI – це звя’зок користувача з системою, який сильно впливає на UX. І для його успішної реалізації варто дотримуватись певних правил:

1. Спрощення взаємодії користувача з застосунком. Чим швидше користувач досягає своєї цілі, ти кращим є користувацький досвід. Як умога менше інтеракцій для досягнення поставленної цілі – ключ до успіху.
2. Колір – є функціональною частиною застосунку. Вибір кольору пов’язаний на взаємодії користувача з системою. Колір допомогає акцентувати увагу, виділяти відмінність між певними елементами, а не тільки бути інструментом естетики.
3. Діалоговий дизайн – є інструментом отримання інформації за допомогою імітації спілкування. Ця техніка може бути набагато кращим методом отримання даних ніж форми.

Ринок мобільних застосунків розвивається і пристрої дають все більше можливостей розробникам та дизайнерам створювати найкращі методи взаємодії застосунку і користувача.

Для організації контролю відвідувань та оцінювання студентів були протестовані наступні мобільні застосунки, основна ідея яких дати можливість організації інформації про навчальний процес.

Кожен мобільний застосунок має свої переваги та недоліки. Вони мають різні функціональні можливості, дизайн і керують різною кількістю даних.

Застосунки оцінюваються за доступними функціональними можливостями та його якістю, красивим та інтуїтивним дизайном UI у відповідності до патернів Material та Cupertino, і ефективним та простим UX.

Підрахована кількість кроків взаємодії, необхідних для досягнення кожної мети, визначеної для кожного сегмента. Проаналізовано, як кількість етапів та складність завдання впливає на користувача.

Одні створені для шкільних вчителів, інші для викладачів університетів. Одні виділяють головні функціональні можливості тільки в якості контролю відвідувань студентів, інші мають комплексний підхід і включають множинний функціонал, котрий включає контроль відвідувань, оцінювання, розклад і навіть статистику.

Деякі застосунки мають непогані UI/UX рішення і виглядають не дуже складними в застосуванні, але здебільшого вони мають перевантажений інтерфейс і відлякують потенційного користувача.

Детальний розгляд і ретельний аналіз допомогли виявити вдалі та погані рішення. За допомогою даних отриманих з цього аналізу було знайдено баланс між найсучаснішими рішеннями UI/UX проблем та найголовнішими функціональними можливостями даного застосунку.

Більшість застосунків можна завантажити з амереканського сервісу цифрового розповсюдження “Play Market”, що керується і розроблений Google. Даний сервіс доступний для усіх мобільних телефонів з операційною системою Android за замовчуванням, при наявності активованого Google акаунту.

2.2 Застосунок «Class Register»

Застосунок «Class Register» можна завантажити з амереканського сервісу цифрового розповсюдження «Play Market» і запустити на операційній системі Android.

При першому запуску, застосунок зображає сторінку від якої пропонується перейти до авторизації та реєстрації натиснувши єдину кнопку на екрані, рисунок 2.1. Дизайн першої сторінки примітивний, а кнопка для переходу до наступного екрану вважається непотрібною.

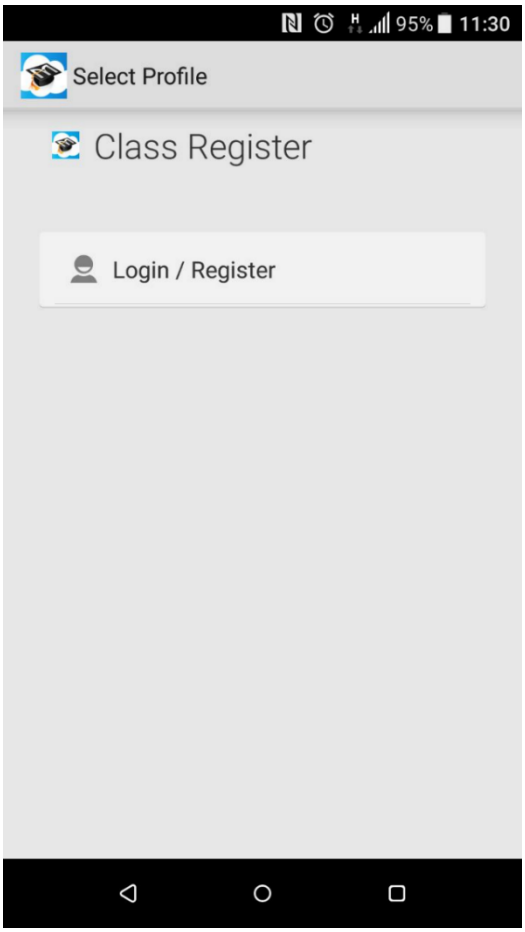

Рисунок 2.1 – Перша сторінка застосунку «Class Register»

Для авторизації пропонується сторінка зі стандартною формою для заповнення, а саме поле електронної пошти та пароль, що є зручно та просто, рисунок

2.2. Також, наявна можливість переходу до сторінки реєстрації. Окрім того, є змога відновити пароль через електронну пошту в разі його втрати.

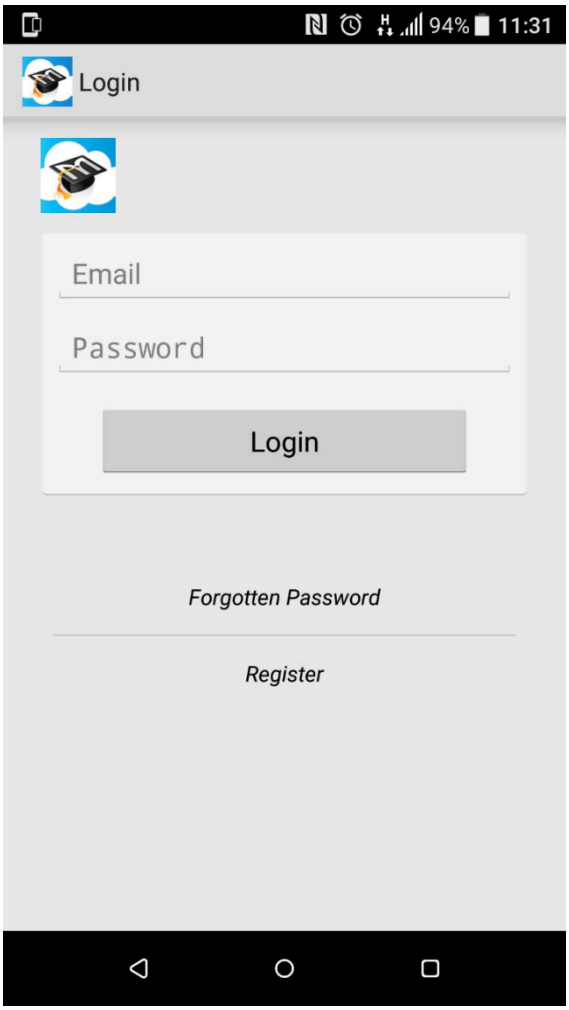

Рисунок 2.2 – Сторінка авторизації застосунку «Class Register»

Для реєстрації пропонується форма з полями імені, прізвища, електронної пошти та паролю, рисунок 2.3.

Після реєстрації користувач відразу переходить до головної сторінки, а саме меню додатку для подальшої навігації. Хоч лист з посиланням для підтвердження створеного акаунту прийшов на електронну скриньку, про це не повідмоляється користувача і він має доступ до застосунку без підтвердження, що лишає його без доступу до деяких функціональних можливостей.

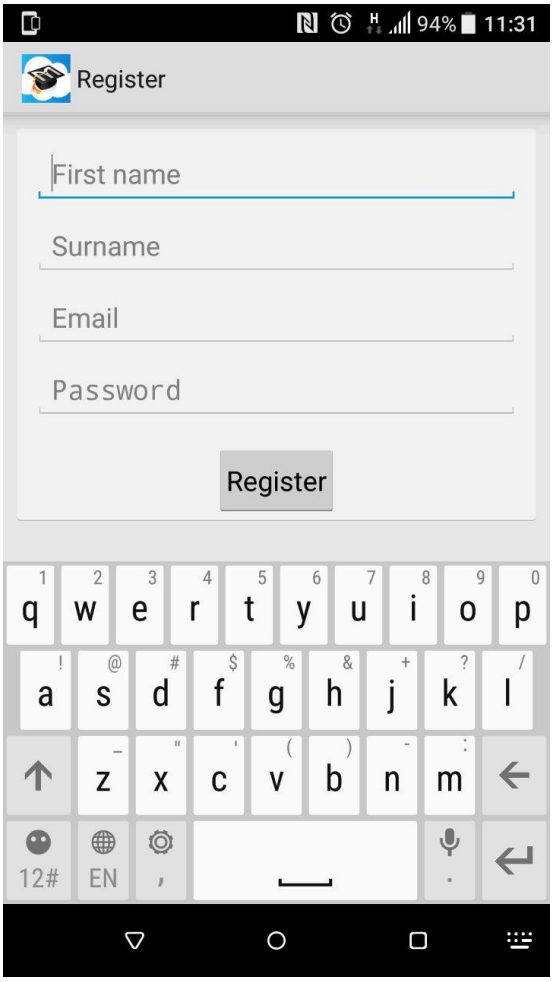

Рисунок 2.3 – Сторінка реєстрації «Class Register»

Меню виглядає багатофункціональним на перший погляд, хоч є доволі перевантаженим, рисунок 2.4.

Після переходу до сторінки найголовніших функціональних можливостей, а саме контролю відвідування студентів, рисунок 2.5, застосунок знову вимагає інформацію для авторазиції до акаунту, тобто адресу електронної скриньки та пароль, рисунок 2.6. Але ця сторінка не доступна, без спеціального сигналу з аудиторії. Інші опції головного меню теж не дають доступу до функціональної можливості створення будь-яких сутностей та контролю відвідування студентів.

Висновок від аналізу даного застосунку – цей програмний продукт не є незалежною системою, що дає змогу будь-якому користувачу в будь-який час та в

будь-якому місці отримати доступ до функціональних можливостей оцінювання та організації контролю відвідувань студентів. Окрім того, застосунок порушує декілька патернів UI/UX, що ще більше відштовхує потенційого користувача.

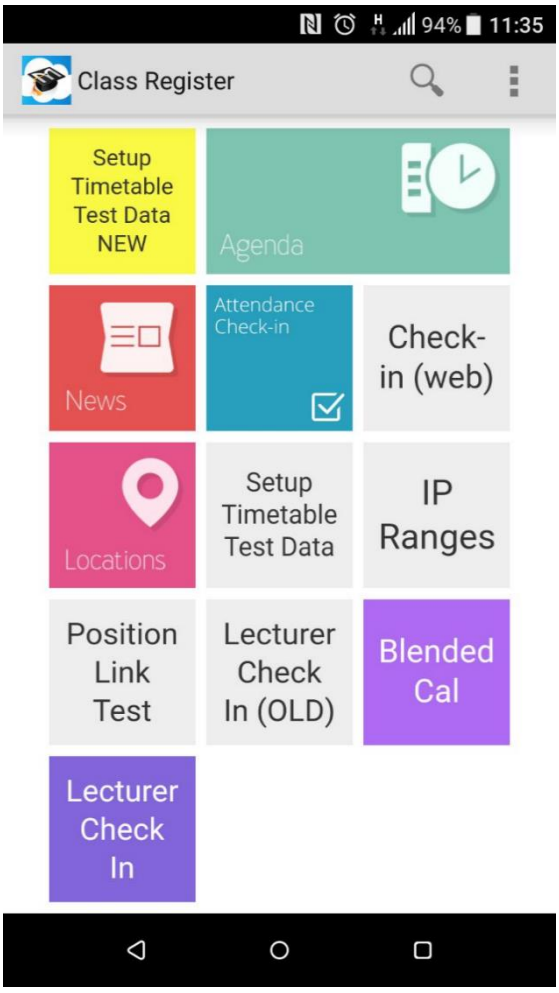

Рисунок 2.4 – Сторінка головного меню застосунку «Class Register»

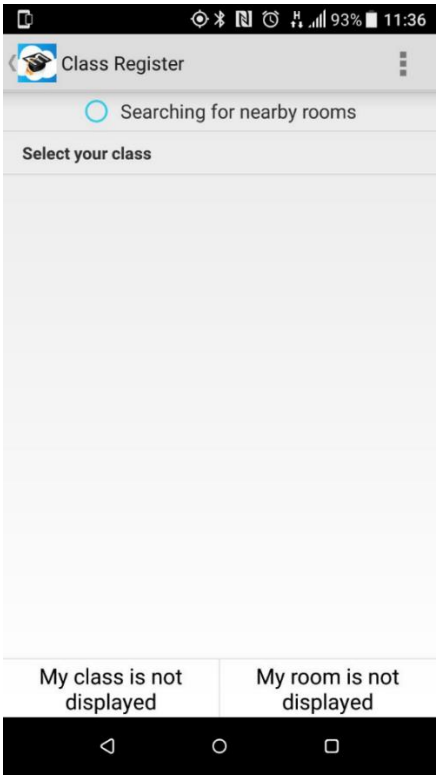

Рисунок 2.5– Сторінка контролю відвідувань студентів застосунку «Class Register»

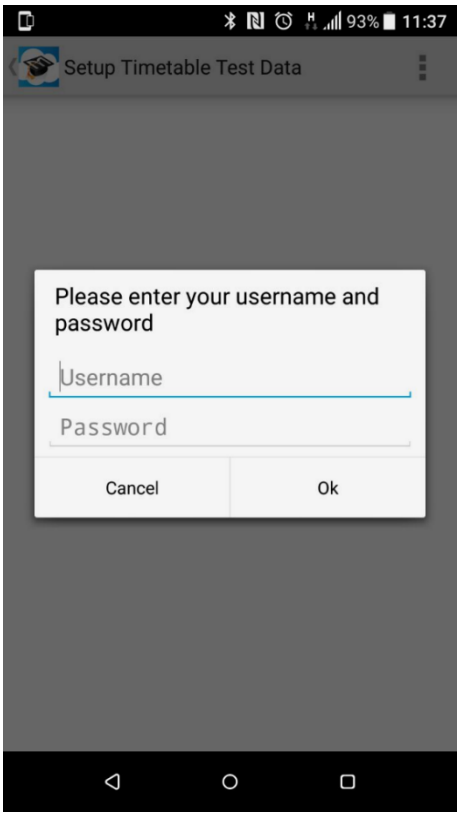

Рисунок 2.6 – Модальне вікно форми авторизації при переході до цієї сторінки застосунку «Class Register»

2.3 Застосунок «Easy Attendance»

Застосунок «Easy Attendance» можна завантажити з амереканського сервісу цифрового розповсюдження «App Store» і запустити на операційній системі iOS.

При запуску, застосунок зображає сторінку авторизація від якої також пропонується перейти до реєстрації натиснувши кнопку «Register now» на екрані, рисунок 2.7. Для авторизації пропонується стандартною формою для заповнення, а саме поле електронної пошти та пароль. Окрім того, є змога відновити пароль через електронну пошту в разі його втрати.

Сторінка реєстрації має стандартні поля повного імені користувача, його електронної пошти, та пароль і підтвердження паролю, що є хорошим прикладом гарного UX, рисунок 2.8.

Варто зазначити, що застосунок пропонує швидку реєстрацію за допомогою вже існуючих акаунтів «Google» та «Facebook», що є дуже зручним і швидким способом почати користувацьку сесію при цьому не створюючи окремий акаунт.

Представлені сторінки мають красивий дизайн та мають зручні в користуванні функціональні можливості.

Після авторизації користувач має змогу користуватися навігацією застосунку за допомогою нижньої панелі навігації, що є стандартним, проте зручним методом.

Одна з вкладок на панелі навігації називається «Add/View», рисунок 2.9, і дає змогу користувачу додати новий курс певного предмету, рисунок 2.10, якщо він натисне кноку «Add Course».

У вкладці контролю відвідування «Take» та у вкладці «Add/View» з’являється створений курс, рисунок 2.11. Якщо користувач натискає на поле з іменем курсу у вкладці «Add/View», з’являється сторінка курсу, де він може додати студентів заповнивши окрему форму.

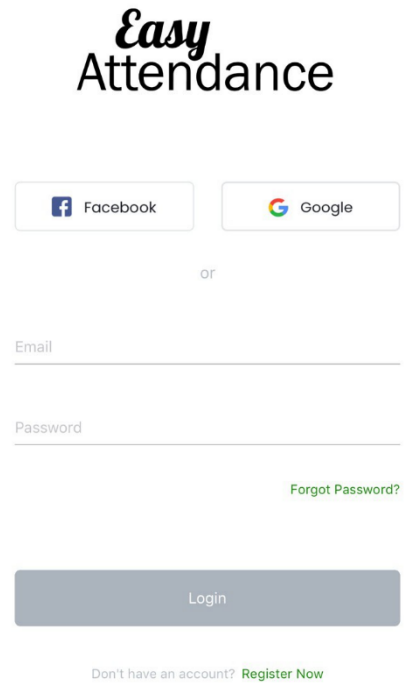

Рисунок 2.7 – Сторінка авторизації застосунку «Easy Attendance»

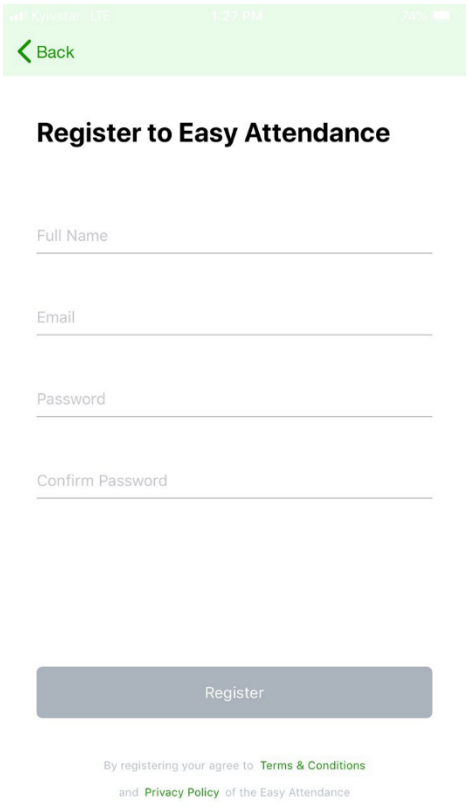

Рисунок 2.8 – Сторінка реєстрації застосунку «Easy Attendance»

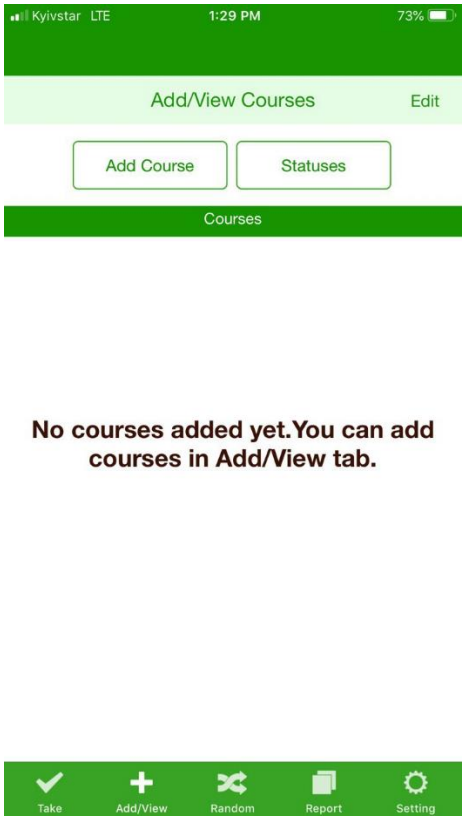

Рисунок 2.9 – Сторінка додавання курсів застосунку «Easy Attendance»

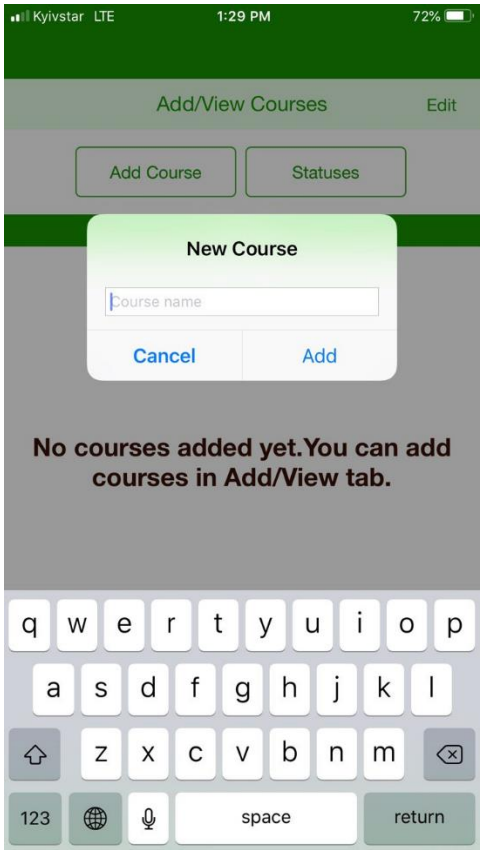

Рисунок 2.10 – Сторінка перегляду курсів застосунку «Easy Attendance»

Якщо користувач натискає на поле з іменем курсу у вкладці «Take», він має обрати дату, коли буде проведений контроль відвудування, рисунок 2.12. Після цього, користувач опиняється на сторінці контролю відвідування і має змогу відмітити кожного студента, що належить до цього курсу, рисунок 2.13.

На перший погляд застосунок «Easy Attendance» виглядає непогано, користувацький інтерфейс не є перевантаженим і має зручнічший користувацький досвід, ніж «Class Register». Проте, варто зазначити, що в даному застосунку зроблено багато UI помилок, наприклад верхня панель управління пуста, а замість неї створенно окрема додаткова панель управління. Програма має широкі функціональні можливості, проте вони не є дуже потрібними і важко доступними для користувача.

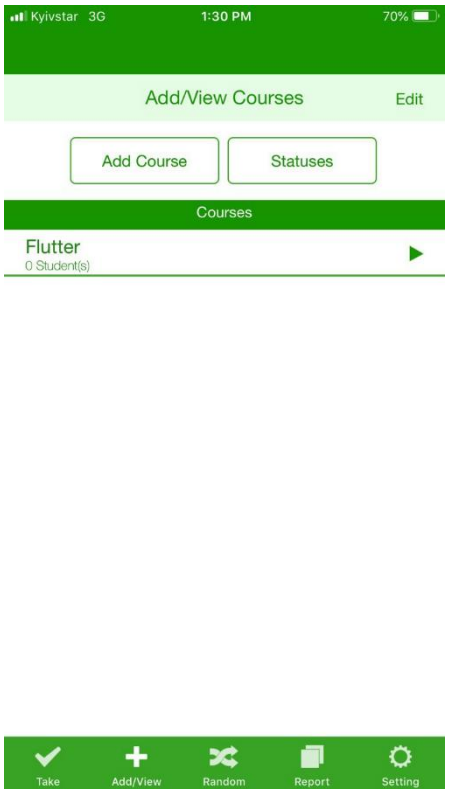

Рисунок 2.11– Список створених курсів застосунку «Easy Attendance»

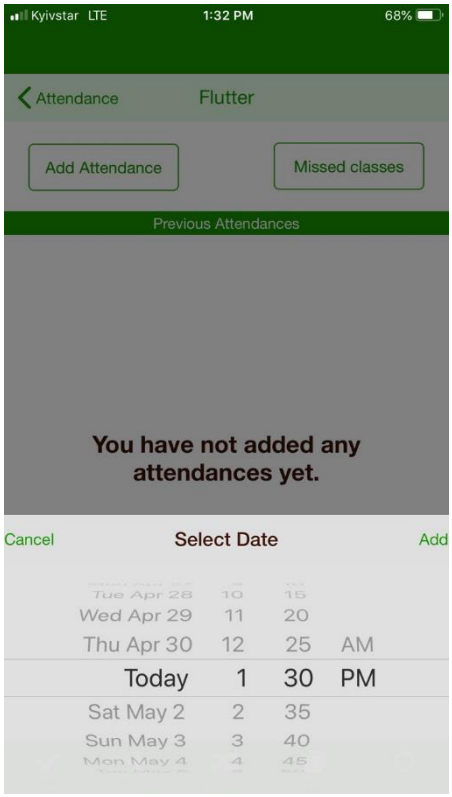

Рисунок 2.12 – Інтерфейс вибору дати проведення контролю відвідування застосунку «Easy Attendance»

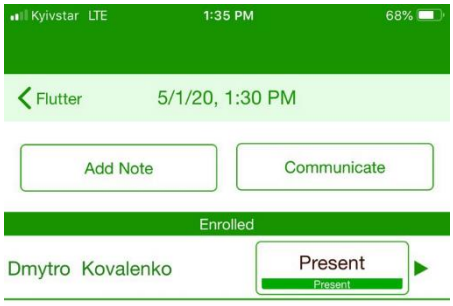

Рисунок 2.13 – Сторінка контролю відвідування застосунку «Easy Attendance»

2.4 Застосунок «Attendance taker»

Застосунок «Attendance taker» можна завантажити з амереканського сервісу цифрового розповсюдження «Play Market» і запустити на операційній системі Android.

При запуску, застосунок відразу зображає головну сторінку, з трьома вкладками. Тобто, всі дані заповнені користувачем зберігаються в фізичному пристрої і немає можливості доступу до них з іншого пристрою. Головна сторінка має верхню панель навігації по трьох доступних вкладках, що є менш зручнішим способом навігації, ніж нижня панель навігації, рисунок 2.14.

В першій вкладці під назвою «Class» користувач має доступ до списку створених курсів студентів та опції їх створення. Натиснувши на кнопку додавання курсу, в центрі нижньої частини екрану. Користувач має ввести назву курсу в модальне вікно.

Після цього, в списку курсів з’являється UI віджет з назвою нового курсу. Якщо натиснути на цей віджет, застосунок відкриває екран редакції курсу і дозволяє додати студента до списку студентів цього курсу. Алгоритм додавання нового студента до списку студентів обраного курсу ідентичний алгоритму додавання нового курсу.

У вкладці під назвою «Attendance» користувач має змогу обрати потрібний курс, натиснувши на віджет з його назвою, рисунок 2.15. Після цього, користувач опиниться на сторінці контролю відвідування, де він має змогу відмітити пристуніх студентів, рисунок 2.16.

Після проведення контролю відвідувань студентів, користувач має натиснути на кнопку збереження редактованої інформації, щоб усі дані були локально збережені.

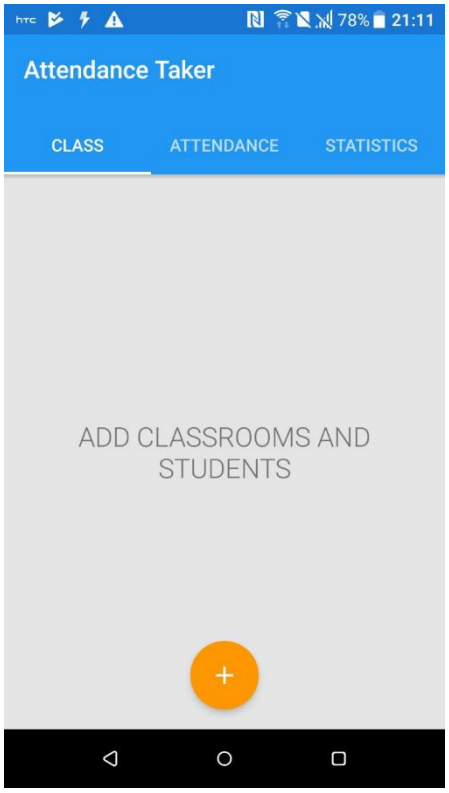

Рисунок 2.14 – Головна сторінка застосунку «Attendance taker»

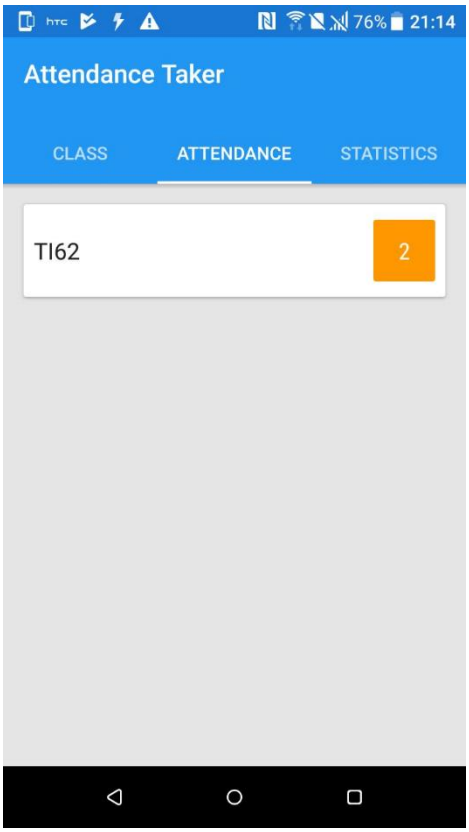

Рисунок 2.15 – Вкладка «Attendance» зі списком існуючих груп застосунку «Attendance taker»

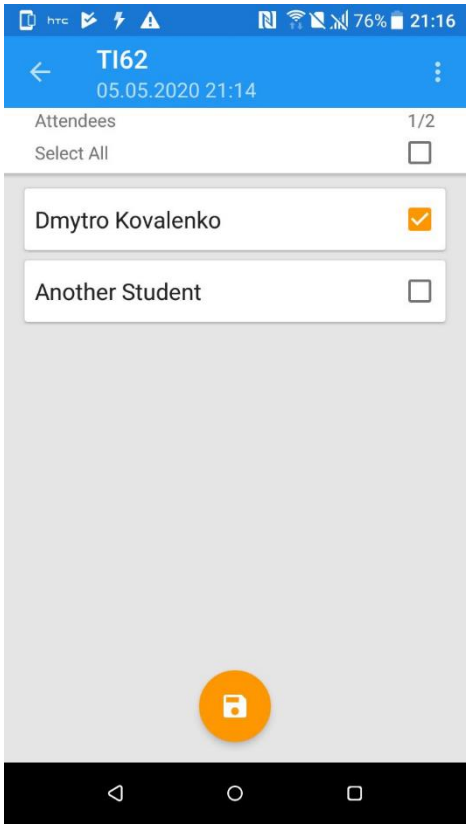

Рисунок 2.16 – Сторінка контролю відвідування студентів, що належать до обраної групи застосунку «Attendance taker»

2.5 Висновки до розділу

У розділі розглянуті основні UX / UI проблеми і важливість різних функціональних можливостей для досягнення поставленої мети.

Були розглянуті існуючі мобільні застосунки, що є аналогами для вирішення задачі. Застосунки «Class Register», «Easy Attendance», «Attendance taker» мають різні методи розв’язання проблеми.

В результаті аналізу вище згаданих застосунків були виділені основні функціональні можливості та ідеї налаштування користувацького досвіду для кращого мобільного застосунку.

При створенні програмного продукту були проаналізовані аналоги і визначена цільова аудиторія. На основі результатів обробки цих даних, було вирішено яким саме побудувати мобільний застосунок.

Мобільний застосунок має бути зручним і можливим для використання в будь-яких навчальних закладах. Проте, він націлений на форму навчання вищих навчальних закладів.

Зазвичай кожний предметний курс має лекційні та практичні сесії. При цьому лекційні сесії певного курсу проводять водночас для всіх груп, що мають в навчальному плані цей курс. В той час як практичні сесії проводяться окремо для кожної групи. Окрім цього, є досить поширеною практикою, коли лектор і викладач на практичних сесіях – це дві різні людини, або лектор може бути викладачем на практичній сесії тільки для окремих груп.

Враховуючи можливі різноманітні варіанти курсів для викладачів, була надана можливість створювати лекційні та практичні курси з можливістю налаштування типу курсу відповідно до кожної групи. Таким чином, визначається чи буде цей курс тільки лекційний, або тільки практичний, або і лекційний, і практичний. За замовчуванням курс для кожної групи є лекційним та практичним.

При створенні функціональних можливостей контролю успішності студентів, було вирішено надати якомога більше свободи користувачу. Важливо, щоб користувач не був обмежений навчальним розкладом або обов’язковими, для заповнення, формами.

Для досягнення зручного досвіду користувача, була нада функціональна можливість створювати навчальні сесії в будь-який момент, не витрачаючи час викладача на пошуки потрібної дати, щодо навчального розкладу, чи заповнення обов’язкової форми для створення учбової сесії. Дата визначається автоматично, а надання назви навчальній сесії опціональна дія. В будь-якому випадку створюється нумерація проведених навчальних сесій, а всі дані яких можна завжди змінити.

3 НАБІР ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ UI «FLUTTER» ТА BAAS FIREBASE

Для виконання поставленої задачі був використаний Flutter – набір інструментів для розробки програмного забезпечення з відкритим кодом, створений Google.

3.1 Flutter

Flutter - це набір інструментів від Google для створення користувацького інтерфейсу для створення красивих програм, створених для мобільних пристроїв, вебу та настільних комп’ютерів із єдиної бази коду. [3]

Головними перевагами Flutter виділяють:

- Користувацький інтерфейс має швидкий цикл створення
- Можливе творення гнучкого та виразного інтерфейсу корисутвача
- Продуктивність нативних застосунків операційних систем iOS та Android не вище застосунків створених за допомогою Flutter

Архітектура фреймворку складається з декількох частин.

Основні компоненти Flutter включають:

- Платформа Dart
- Двигун Flutter
- Основна бібліотека
- Віджети, характерні для певних дизайнів

Набір інструментів Flutter написаний мовою Dart і успішно використовує багато вдосконалених функціональних можливостей цієї нової мови.

У Windows, macOS та Linux за допомогою офіційного проекту Flutter Desktop Embedding Flutter працює у віртуальній машині Dart, яка має функціональний механізм виконання. Під час написання та налагодження програми Flutter використовує компіляцію Just In Time, що забезпечує «гаряче перезавантаження», за допомогою чого модифікації вихідних файлів можна вводити в запущену програму. Flutter поширює це на підтримку стаціонарного «гарячого перезавантаження», де в більшості випадків зміни вихідного коду можуть бути відображені негайно в запущеному додатку, не вимагаючи перезавантаження або втрати стану. Ця функція, реалізована у Flutter, отримала широку позитивну оцінку.

Версії випуску програм Flutter складені за допомогою дострокової (AOT) компіляції для Android та iOS, завдяки чому можливі високі показники роботи Flutter на мобільних пристроях. [4]

Двигун Flutter, написаний головним чином на C ++, забезпечує підтримку низькорівневого візуалізації за допомогою графічної бібліотеки Skia від Google. Крім того, він взаємодіє з платформою, орієнтованою на платформу SDK, такими, які надаються Android та iOS. [4] Flutter engine - має портативні програми для розміщення програм Flutter. Він реалізує основні бібліотеки Flutter, включаючи анімацію та графіку, введення / виведення файлів та мереж, підтримку доступності, архітектуру плагінів та набір інструментів Dart. Більшість розробників взаємодіють з Flutter через Flutter Framework, який забезпечує сучасний, чуйний фреймворк та багатий набір віджетів платформи, верстка та фундамент.

Основна бібліотека, написана мовою програми Dart, містить основні класи та функції, які використовуються для побудованої програми Flutter, як API для комунікації з двигуном Flutter. [4]

Набір інструментів Flutter містить два набори сегментів, які виконують конкретну дизайнерську конструкцію. Використовуючи Material Design, розробник явно використовує патерни дизайну від Google, якщо ж Cupertino, то патерни дизайну від Apple, відомі під назвою Apple iOS Human Interface. [3]

Дизайн інтерфейсу у Flutter забезпечує виконання композицій для розробки / створення застосунку з іншими відтінками. Раніше у виробництві було хитрістю використовувати те, що будь-які компоненти мали деревоподібну структуру, зібрані в рамках одного методу build (), також називаються одинарним відтворенням. Більше того, ці невеликі результативні віджети складаються з ще менших результативних віджетів, і у кожного з них є власний метод build (). Ось як Flutter використовує компіляцію. [8]

Згідно з документації: "Віджет - це незмінний опис частини інтерфейсу користувача". Логіка цього, як у креслення, це найкращий спосіб зрозуміти його сенс. Однак потрібно пам’ятати, що у Flutter існує багато різних типів віджетів, і їх не можна побачити або торкнутися на екрані інтерфейсу. Текст є віджетом, але також його стиль теж є віджетом, який визначає такі речі, як розмір, колір, сімейство шрифтів та вага. Є віджети, які представляють речі, такі, які представляють характеристики (наприклад, TextStyle) і навіть інші, які роблять різні речі, наприклад FutureBuilder або StreamBuilder.[9]

Складні віджети можна створити, поєднавши безліч простіших, і весь застосунок є лише найбільшим із усіх віджетом (часто має назву "MyApp"). Віджет MyApp містить усі інші віджети, які можуть містити ще менші віджети, і разом вони складають ваш додаток.

Однак використання віджетів не є обов’язковою і необхідною умовою для створення програм Flutter. Альтернативний варіант, що зазвичай використовується та застосовується лише людьми, які вважають за потрібне контролювати кожен піксель, створений на екрані, - це можливо, якщо безпосередньо використовувати методи основної бібліотеки. Ці методи можна використовувати для малювання

фігур, тексту та зображень безпосередньо на екрані. Ця здатність Flutter була використана в декількох фреймворках, таких як ігровий двигун з відкритим кодом Flame.[8]

3.2 Dart

Dart – це клієнто-оптимізована мова програмування для створення застосунків на разних платформах. Розроблена компанією Google, вона використовується для створення мобільних, комп'ютерних, серверних та веб застосунків. [3]

Варто зазначити, що мова Dart базується на класах і є об'єктно-орієнтованою мовою, що має вбудовану технологію «garbage collector» і має синтаксис подібний до мови C. Ця мова може компілюватися в нативний код чи JavaScript. Мова програмування Dart підтримує можливість створення інтерфейсів, абстракних класів, комбінацій, розширену шаблонізацію, та умовививоди.

Існує чотири способи виконання коду написаного мовою Dart:

- Скомпільовано як JavaScript

  Для запуску в основних веб-браузерах Dart покладається на компілятор від джерела до джерела в JavaScript. За даними сайту проекту, Dart був «розроблений так, щоб легко писати інструментарії для розробки, чудово підходять до сучасної розробки застосунків і здатні до високопродуктивної реалізації в проектах різних масштабів». [3] Під час запуску коду Dart у веб-браузері код попередньо компілюється у код мови JavaScript за допомогою компілятора dart2js. Скомпільований як JavaScript, Dart-код сумісний з усіма основними браузерами, у яких браузери не потребують прийняття Dart. Завдяки оптимізації скомпільованого виводу JavaScript, щоб уникнути дорогих перевірок і операцій, код, записаний мовою Dart, може в деяких випадках

працювати швидше, ніж еквівалентний код, написаний вручну, використовуючи ідіоми JavaScript. [3]

- Автономно

Інструментарій розробки програмного забепечення Dart (SDK) постачається з автономною програмою Dart VM, що дозволяє виконувати код мови Dart в інтерфейсі командного рядка. Оскільки мовні засоби, включені до SDK Dart, написані здебільшого на Dart, автономна Dart VM є важливою частиною SDK. Ці інструменти включають компілятор dart2js та менеджер пакунків під назвою pub. Dart постачається з повною стандартною бібліотекою, що дозволяє користувачам писати повністю працюючі системні застосунки, такі як користувацькі веб-сервери. [5]

- Скомпільовано Ahead-of-time

Програмний код мови програмування Dart можна компілювати AOT у машинний код (нативний набір інструкцій). Програми, створені за допомогою Flutter, SDK для мобільних застосункуів, створених за допомогою мови Dart, розміщуються в магазинах застосунків як AOT, що є скомпільованим кодом Dart. [5]

- Нативно

Мова програмування Dart може бути скомпільована в нативний код для запуску мобільних застосунків на пристроях з операційними системами iOS та Android через Flutter. [3]

Додаткові пакети з бібліотеками та утилітами розповсюджуються через сховище pub, яке станом на весну 2015 року має понад півтори тисячі пакунків, включаючи фреймворки для розробки веб-додатків AngularDart та polymer.dart. [5]

Особливості мови Dart:

• знайомий і простий у вивченні синтаксис, природний для програмістів у JavaScript, C та Java;

• забезпечення швидкого запуску та високої продуктивності для всіх сучасних веб-браузерів та різних типів середовищ, від портативних пристроїв до потужних серверів;[10]

• Можливість визначення класів та інтерфейсів, що дозволяють використовувати інкапсуляцію та повторно використовувати існуючі методи та дані;

• Додаткові інструкції щодо типу, незалежно від того, використовувати статичні типи чи ні, залежить від розробника. Визначення типів полегшує налагодження та ідентифікацію помилок, робить код більш зрозумілим та читаним, спрощує його вдосконалення та аналіз сторонніми розробниками. [10]

• Серед підтримуваних типів: різні типи хешей, масивів і списків, черг, числових і рядкових типів, типи визначення дати та часу, регулярні вирази (RegExp). Можна створити власні типи; [10]

• для організації паралельного виконання пропонується використовувати класи з атрибутом isolate, код яких працює повністю в ізольованому просторі в окремій області пам’яті, взаємодіючи з основним процесом за допомогою надсилання повідомлень;

• Підтримка використання бібліотек, які спрощують обслуговування та налагодження великих веб-проектів. Іноземні реалізації функцій можуть бути підключені у вигляді спільних бібліотек. Програми можна розділити на частини, і кожна розробка деталей може бути віднесена до окремої команди програмістів; [10]

• Набір готових інструментів для підтримки розвитку мовою Dart, включаючи реалізацію динамічних інструментів розробки та налагодження з виправленням коду на ходу ("редагування та продовження"). [10]

• Можливість створення однорідних систем, що охоплюють як клієнтські, так і серверні частини. Використовуйте унікальну мову та інструменти для

клієнтських та серверних компонентів, які намагаються переробити процес, а також зазнають змін у контексті.

3.3 Firebase

Firebase - це платформа для розробки мобільних і веб-додатків, розроблена Firebase, Inc. у 2011 році, потім придбана Google у 2014 році. Станом на березень 2020 року платформа Firebase налічує 19 продуктів [6] , які використовуються понад 1,5 мільйона додатків.

Сервіси Firebase включають:

- Google Analytics
    - Google Analytics - це безкоштовне рішення для вимірювання додатків, яке забезпечує розуміння використання програми та залучення користувачів. [6]
- Firebase Cloud Messaging
    - Раніше відомий як Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) - це крос-платформене рішення для повідомлень та сповіщень для Android, iOS та веб-додатків, яке станом на 2016 рік можна використовувати безкоштовно. [6]
- Firebase Authentication
    - Автентифікація Firebase - це послуга, яка може аутентифікувати користувачів, використовуючи лише код на стороні клієнта. Він підтримує провайдерів соціальних входів Facebook, GitHub, Twitter та Google, а також інших постачальників послуг, таких як Google Play Games, Apple, Yahoo та Microsoft. Крім того, вона включає систему управління користувачем, за допомогою якої розробники

можуть вмикати автентифікацію користувача за допомогою електронної пошти та входу пароля, що зберігаються з Firebase. [6]

- Firebase Realtime Database
    - Firebase надає базу даних в реальному часі та бек-енд-сервіс як послугу. Служба надає розробникам додатків API, який дозволяє синхронізувати дані застосунків між клієнтами та зберігати їх у хмарі Firebase. Компанія надає клієнтські бібліотеки, що забезпечують інтеграцію з застосунками Android, iOS, JavaScript, Java, Objective-C, Swift та Node.js. База даних також доступна через API REST та прив’язки для декількох фреймворків JavaScript, таких як AngularJS, React, Ember.js та Backbone.js. API REST використовує протокол Server-Sent Events, що є API для створення HTTP-з'єднань для отримання push-повідомлень від сервера. Розробники, що використовують базу даних в режимі реального часу, можуть захистити свої дані, використовуючи правила безпеки компанії, що застосовуються на сервері. [6]
- Cloud Firestore
    - Cloud Firestore - це швидка, повністю керована без сервера базова база даних NoSQL, яка спрощує зберігання, синхронізацію та запит даних для мобільних додатків, Інтернету та IoT у глобальному масштабі. Клієнтські бібліотеки забезпечують живу синхронізацію та підтримку в режимі офлайн, тоді як його функції безпеки та інтеграції з Firebase та Google Cloud Platform (GCP) прискорюють створення програм без серверів. [6] [7]
    - За допомогою Cloud Firestore програми можуть оновлюватися майже в реальному часі, коли змінюються дані на зворотному кінці. Це не тільки зручно для створення спільних багатокористувацьких мобільних застосунків, але також означає, що є можливість тримати

дані синхронізовано з окремими користувачами, які, можливо, хочуть використовувати застосунок з декількох пристроїв.

- Cloud Firestore має повну офлайн-підтримку, тож є можливість отримати доступ до своїх даних та змінити їх, і ці зміни будуть синхронізовані з хмарою, коли клієнт повернеться в Інтернет. Вбудована підтримка в режимі офлайн використовує локальний кеш для обслуговування та зберігання даних, тому застосунок залишається чутливим незалежно від затримки мережі або підключення до Інтернету.[7]

- Firebase Storage
  - Firebase Storage забезпечує захищене завантаження та завантаження файлів для програм Firebase, незалежно від якості мережі, що використовуються для зберігання зображень, аудіо, відео чи іншого створеного користувачем вмісту. Це підтримується Google Cloud Storage. [6]
- Firebase Hosting
  - Firebase хостинг - це статична та динамічна веб-хостинг-служба, яка запустилася 13 травня 2014 року. Він підтримує розміщення статичних файлів, таких як CSS, HTML, JavaScript та інші файли, а також підтримку за допомогою хмарних функцій. Послуга доставляє файли через мережу доставки вмісту (CDN) за допомогою HTTP Secure (HTTPS) та шифрування шару захищених сокетів (SSL). Firebase співпрацює з CDN швидко, щоб забезпечити підтримку CDN-хостингу Firebase. Компанія заявляє, що Firebase хостинг виріс із запитів клієнтів; розробники використовували Firebase для своєї бази даних у реальному часі, але їм потрібно було місце для розміщення їх вмісту. [6]

- ML Kit
    - ML Kit - це система навчання мобільних машин для розробників, запущена 8 травня 2018 року в бета-версії під час Google I / O 2018. API ML Kit мають різноманітні функції, включаючи оптичне розпізнавання символів, виявлення облич, сканування штрих-кодів, маркування зображень та розпізнавання орієнтирів. Зараз він доступний для розробників iOS або Android. Ви також можете імпортувати власні моделі TensorFlow Lite, якщо даних API не вистачає. API можна використовувати на пристрої або в хмарі. [6]
- Crashlytics
    - Crash Reporting створює докладні звіти про помилки в застосунку. Помилки згруповані в кластери схожих слідів стека і викликаються серйозністю впливу на користувачів застосунків. Окрім автоматичних звітів, розробник може реєструвати власні події, щоб допомогти фіксувати кроки, що ведуть до аварії. [6]
- Performance
    - Ефективність Firebase дає змогу зрозуміти ефективність програми та затримки користувачів, які користувачі програми. [6]
- Firebase Test Lab
    - Тестова лабораторія Firebase забезпечує хмарну інфраструктуру для тестування застосунків Android та iOS за одну операцію. Розробники можуть протестувати свої програми на широкому спектрі пристроїв та конфігурацій пристроїв. Результати тестів - включаючи журнали, відео та скріншоти - доступні в консолі Firebase. Навіть якщо розробник не написав тестовий код для своєї програми, Test Lab може застосувати додаток автоматично, шукаючи збоїв. Тестова лабораторія для iOS наразі знаходиться в бета-версії. [6]
- Admob

- Admob - продукт Google, який інтегрується з аудиторією Firebase. [6]

- Firebase Dynamic Links
  - Посилання Dynamic Firebase - це розумні URL-адреси, які динамічно змінюють свою поведінку, щоб забезпечити "найкращий доступний досвід" на різних платформах, включаючи веб-браузери для настільних ПК, iOS та Android, а також поглиблені посилання на мобільні застосунки. Динамічні посилання працюють у всіх встановленнях додатків: якщо користувач відкриє Dynamic Link на iOS або Android, а програма не встановлена, користувачеві буде запропоновано спочатку встановити застосунок. Після встановлення програма почне працювати та матиме доступ до посилання. [6]

В даній роботі були використані сервіси Firebase, що мають назву Firebase Authentication та Cloud Firestore.

3.4 Висновки до розділу

У цьому розділі були розглянуті використані технології Flutter, Dart та Firebase і їх функціональні можливості.

Також було ознайомлено з основними принципами та специфікаціями на яких будуються вищевказані технології. Розглянуто особливості технологій та представлені їх переваги.

Кожна з представлених технологій використовує найсучасніші методи розробки програмного забезпечення і підкріплюється Google Inc.

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Мобільний застосунок призначений для одного актора – користувача, що має за ціль проконтролювати присутність студентів певних груп на лекції та виставити оцінки за виконання практичної роботи.

4.1 Структура програми

Під час розробки мобільного застосунку була спроектована і реалізована така схема системи мобільного застосунку, рисунок 4.1.

Клієнт-серверний застосунок, при використані котрого користувач отримує дані з одного із сервісів Firebase, назва якого Cloud Firestore. Користувач може отримати доступ до Cloud Firestore, тільки якщо його дані провалідовані правилами сервісу, в яких зазначено, що тільки клієнт отримавший дозвіл Authentication Service, може зчитувати та змінювати дані, що пов’язані з його акаунтом.

При відсутності інтернет підключення, застосунок використовує кеш дані, що зберігаються в пам’яті мобільного пристрою. Всі запити на зміни, що виникають при відсутності інтернет підключення збергаються в кеш пам’яті, і потім, при його появі посилаються до сервісу Firebase.

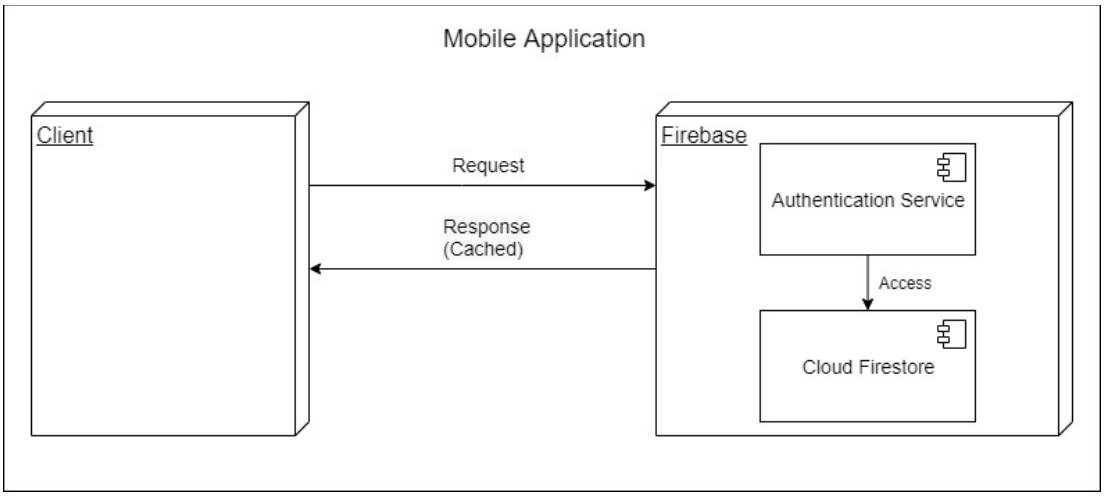

Рисунок 4.1– Схема роботи системи мобільного застосунку

Діаграма класів, рисунок 4.2, демонструє як мобільний застосунок керує даними в масштабі всієї системи. Ця діаграма не репрезентує класи, що відповідають за візуальну складову, тобто класи-віджети, а демонструє класи, що відображають структуру даних і їх робочий потік. Вважається, що діаграма класів-віджетів занадто об’ємна, а їх головні функціональні можливості будуть краще продемонстровані у вигляді діаграми прецедентів.

Клас Model є головним зв'язковим класом, що контрлює всю систему. Використовуючи класи сутності, а саме Student, Group, Practice, Lecture, клас Model керує змінами в активних даних і посилає запит на ці зміни за допомогою класу FirebaseService.

Задача класу FirebaseService – відіслати певний запит до сервісу Firebase та отримати його відповідь, щоб потім відправити дані резльтату запиту на опрацювання класом Model.

Класи сутностей, а саме класи Student, Group, Practice, Lecture мають дві головні задачі. Перша – це представлення даних в зручному вигляді, що дає змогу будувати надійний застосунок і уникати помилок пов’язаних з типізацією даних. Друга – це конвертація даних json файлу, що приходить в якості результату на будь-який запит до Cloud Firestore. Кожен з цих класів, може створити об’єкт від відповідних даних json файлу та перетворити дані об’єкту в структуру json.

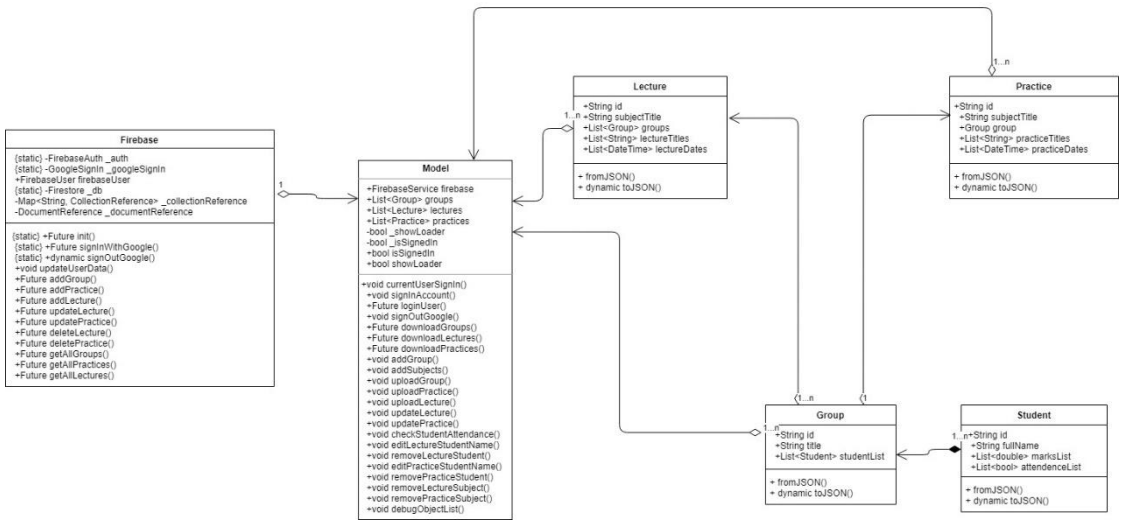

Рисунок 4.2 – Діаграма класів мобільного застосунку

Cloud Firestore – це не реляційна база даних, що зберігає всі необхідні дані користувача. Діаграма структури бази даних демонструє в якому вигляді зберігаються дані, пов’язані з користувачем, рисунок 4.3.

Ця структура відображає, що кожен користувач має список сутностей лекцій, практик та груп, при цьому кожна практика і лекція теж мають групи, а кожна група має список студентів.

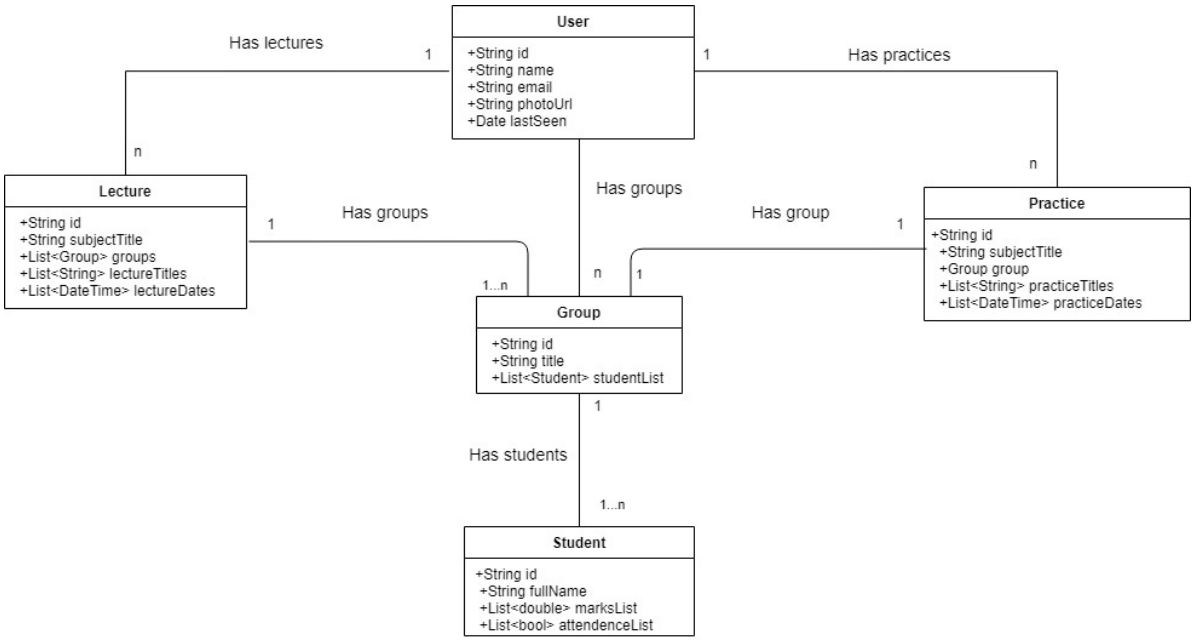

Рисунок 4.3 – Структура бази даних

4.2 Опис розроблених алгоритмів

Для представленя наданих застосунком функціональних можливостей спроектована діаграма прецедентів. Через велику кількість прецедентів в діаграмі, вона була декомпозована на декілька частин.

Дана діаграма зображує відношення між акторами, в конкретному випадку користувачем, та прецедентами в системі, рисунок 4.4. Ця діаграма відображає елементи моделі варіантів використання застосунку. Діаграма прецедентів показує які послуги система надає актору. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

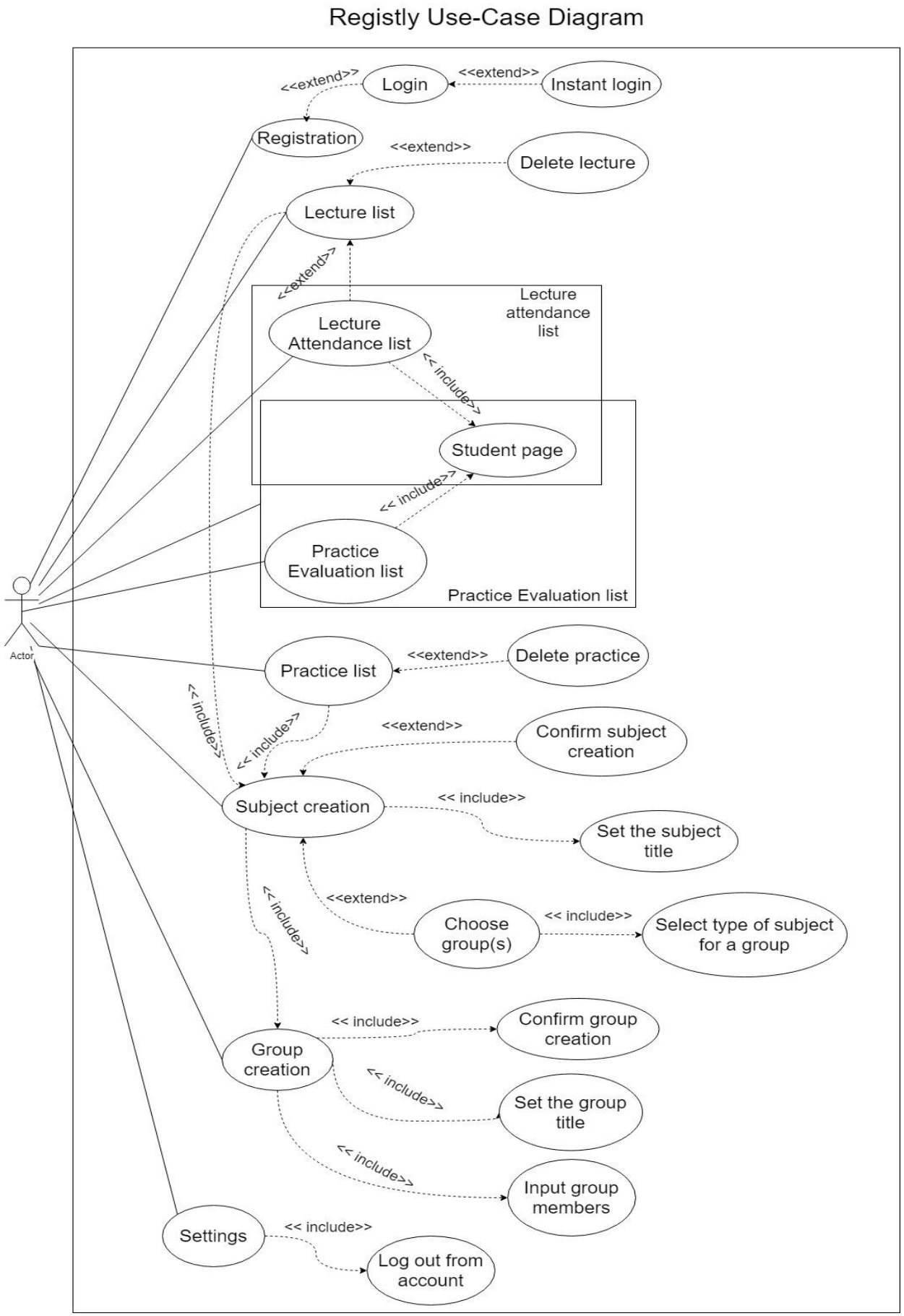

Рисунок 4.4 – Діаграма прецедентів мобільного застосунку

Прецеденти лекції і практики були декомпозовані і представлені на рисунках 4.5 і 4.6.

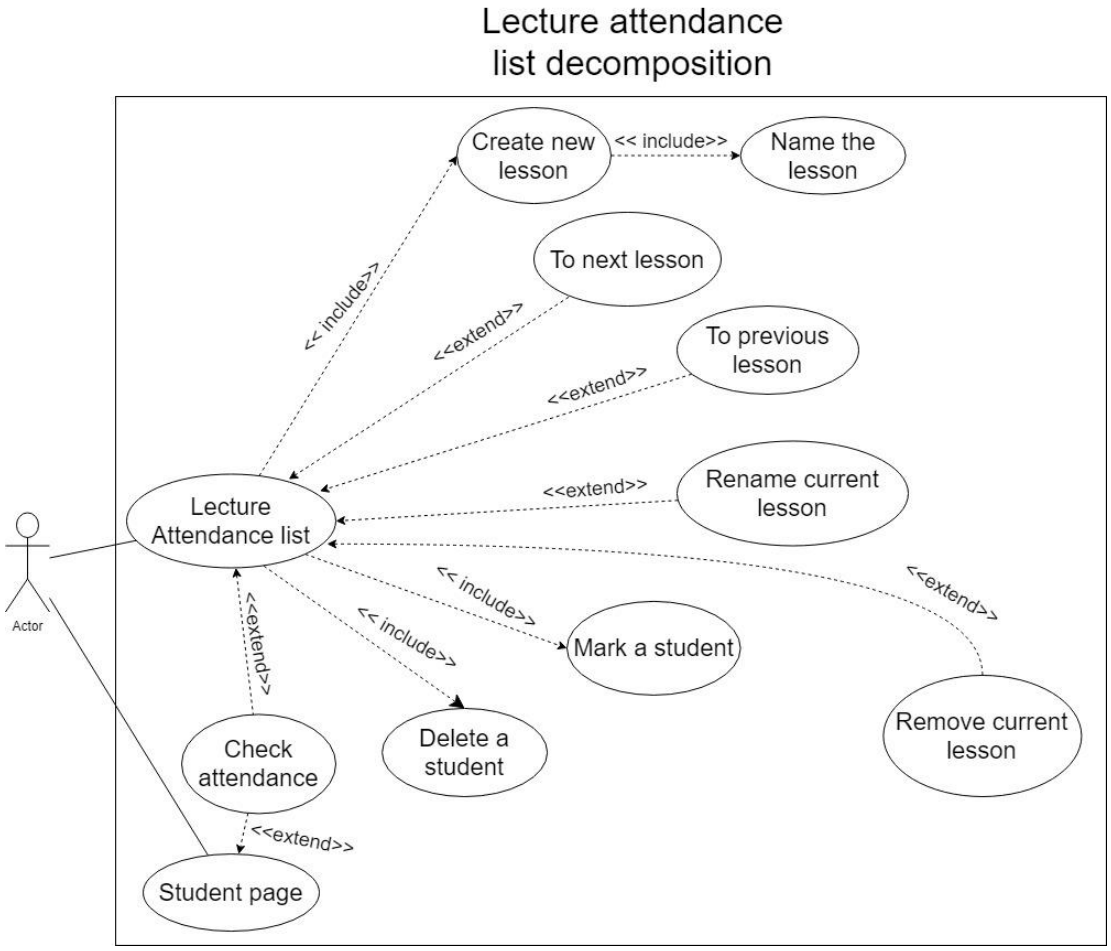

Рисунок 4.5 – Декомпозиція прецеденту списку відвідування лекції

З даних діаграм можна вивести наступні алгоритми дій актора для досягнення своєї цілі.

- Щоб увійти до системи, актор має:

1. Зареєструватись
2. Підтвердити адресу електронної пошти
3. Авторизуватись

Або

1. Авторизуватись використовуючи акаунт Google

- Щоб створити новий лекційний і / або практичний курс, актор має:

1. Визначити назву курсу

2. Обрати групи, що належать до нього
3. Обрати тип курсу для кожної з обраних груп
4. Підтвердити створення курсу

- Щоб створити нову групу, актор має:

1. Визначити назву групи
2. Ввести імена студентів групи
3. Підтвердити створення групи

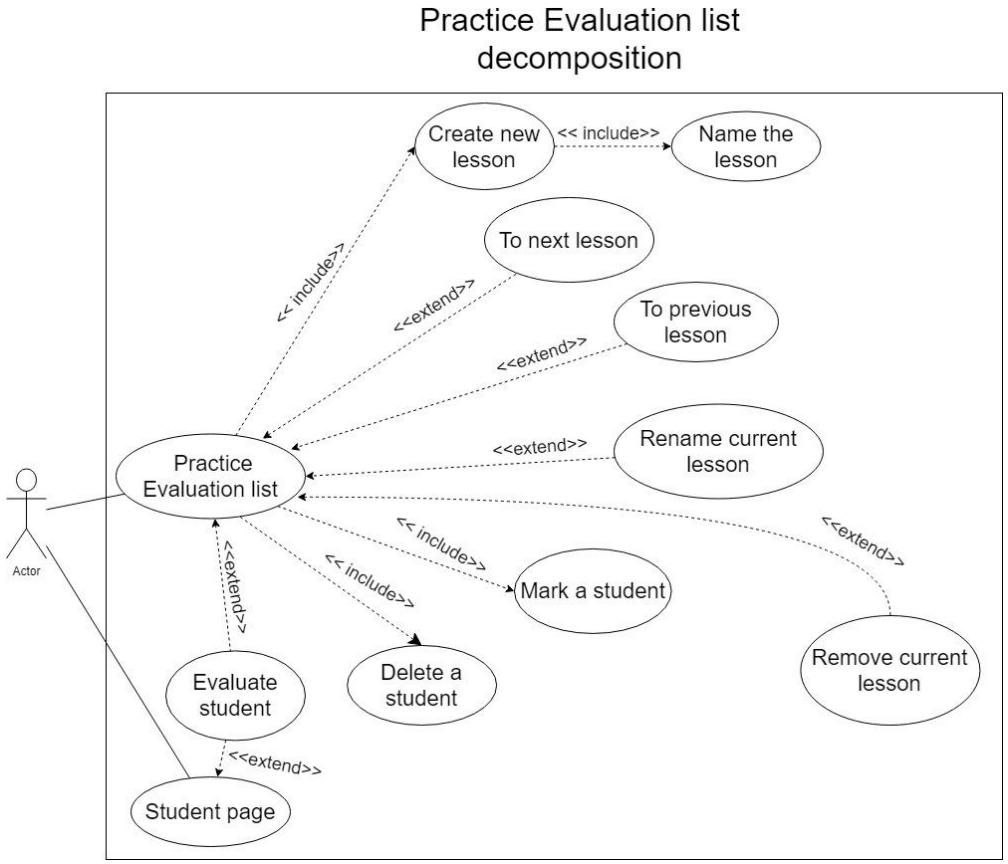

Рисунок 4.6 – Декомпозиція прецеденту списку оцінювання практичної роботи

- Щоб провести контроль відвідувань студентів, актор має:

1. Обрати потрібний лекційний курс
2. Створити потрібну сесію та, за бажанням, визначити її тему
3. Провести контроль відвідувань студентів, обираючи потрібних учнів

- Щоб провести оцінювання практичних робіт студентів, актор має:

1. Обрати потрібний практичний курс
2. Створити потрібну сесію та, за бажанням, визначити її тему
3. Провести провести оцінювання практичних робіт студентів, обираючи потрібних учнів

- Щоб провести контроль відвідування певного студента по всім існуючим сесіям обраного курсу, актор має:

1. Обрати потрібного студента зі списку групи обраного курсу
2. Перейти на сторінку цього студента
3. Провести контроль відвідувань

- Щоб провести оцінювання практичних робіт певного студента по всім існуючим сесіям обраного курсу, актор має:

1. Обрати потрібного студента зі списку групи обраного курсу
2. Перейти на сторінку цього студента
3. Провести оцінювання практичних робіт

4.3 Керівництво користувача

Для того, щоб скористатись мобільним застосунком для організації контролю відвідувань та оцінювання студентів користувач має запустити застосунок на операційній системі Android або iOS.

Коли користувач відкриває застосунок, він має авторизуватись для того, щоб отримати доступ до системи. Даний застосунок має два способи авторизації, рисунок 4.7. Перший – це стандартна авторизація, для проходження якої, потрібно заповнити поля адреси електронної пошти та пароль до існуючого акаунту системи. Другий –

використати існуючий акаунт Google для авторизації, натиснувши на відповідну кнопку на екранному інтерфейсі, рисунок 4.8.

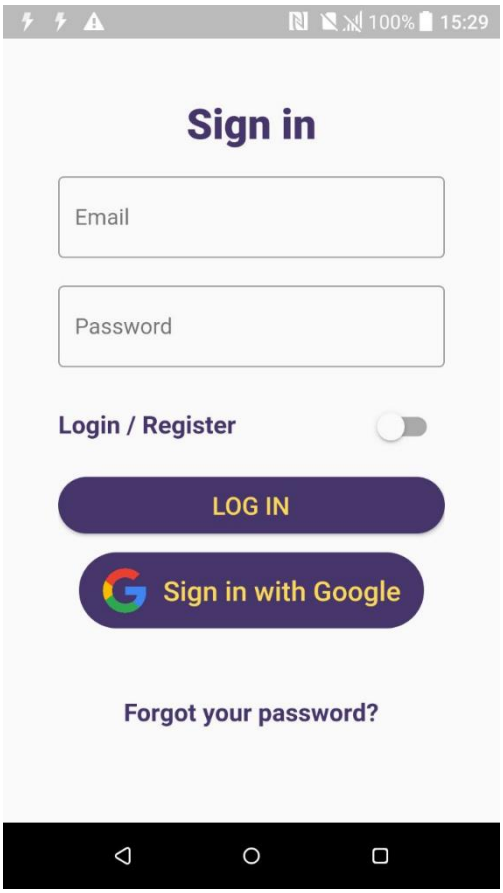

Рисунок 4.7 – Інтерфейс авторизації в систему

Якщо користувач вперше використовує дану систему і не має змогу або бажання використовувати акаунт Google для авторизації, він має змогу пройти реєстрацію ввівши свою електронну пошту та пароль до свого акаунту, рисунок 4.9. Після підтвердження форми реєстрації, користувач має підтвердити адресу своєї електронної пошті, перейшовши по посиланню, яке він має змогу знайти в свої електронній поштовій скриньці. Після підтвердження, користувач має змогу авторизуватися до системи.

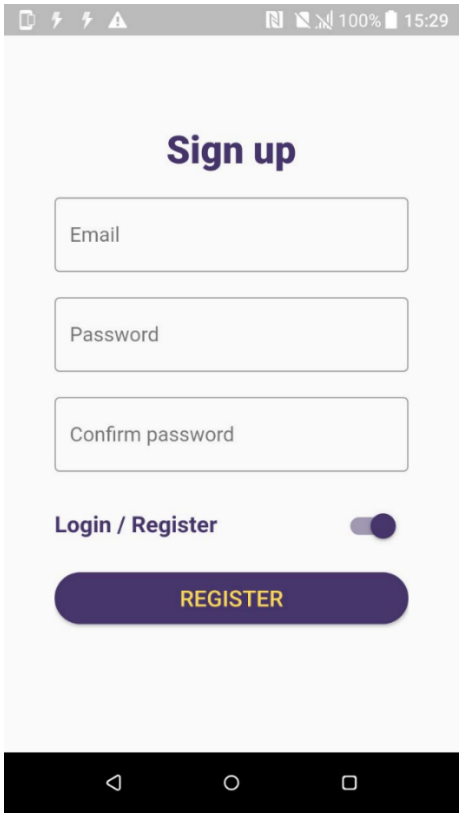

Рисунок 4.8 – Інтерфейс реєстрації в систему

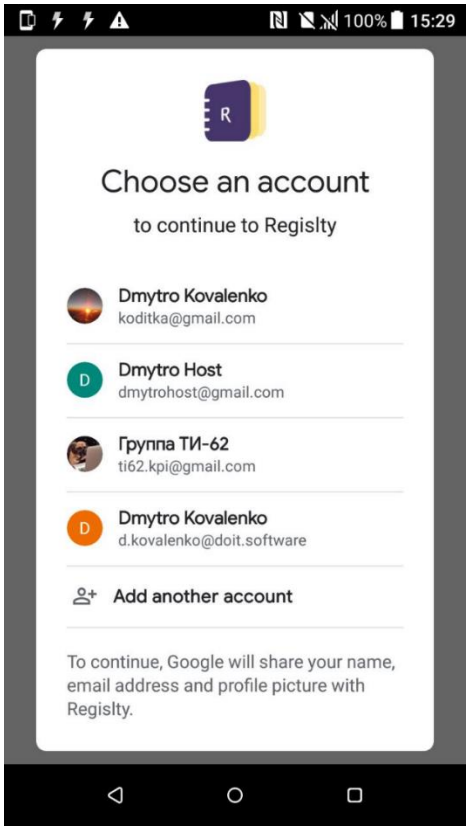

Рисунок 4.9 – Використання акаунту Google для авторизації

У разі втрати паролю до існуючого акаунту системи, користувач має змогу скористуватись можливістю відновлення паролю натиснувши на відповідний текст в нижній частині екранного інтерфейсу, рисунок 4.7. Достатньо ввести адресу електронної пошти і підтвердити відновлення паролю. Після цього користувач отримує лист з посиланням на свою електронну пошту. Перейшовши по посиланню, користувач має змогу заповнити форму для відновлення паролю, ввівши новий пароль. Після цього він має змогу авторизуватись з новим паролем.

Після авторизації, користувач мобільного застосунку опиняється у вкладці списку курсів лекцій, рисунок 4.10. Окрім цієї вкладки, є ще вкладка списку курсів практик та налаштувань. Варто зазначити, що кожна вкладка маю свою власну навігацію і переходячи по ним, історія кожної вкладки зберігається окремо. Ця властивість спеціально розроблялась для зручного користування функціональними можливостями застосунку.

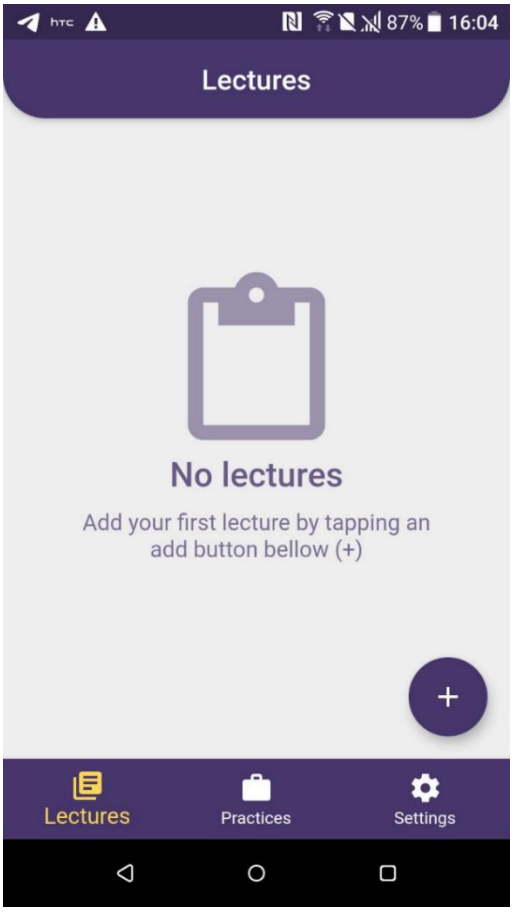

Рисунок 4.10 – Інтерфейс застосунку на вкладці списку лекцій

Щоб додати курс лекцій або практик до списку, достатньо натиснути кнопку зі знаком «+» в нижньому правому кутку інтерфейсу вкладки списку курсів лекцій або практик. Ця дія переносить користувача на сторінку створення курсу, рисунок 4.11.

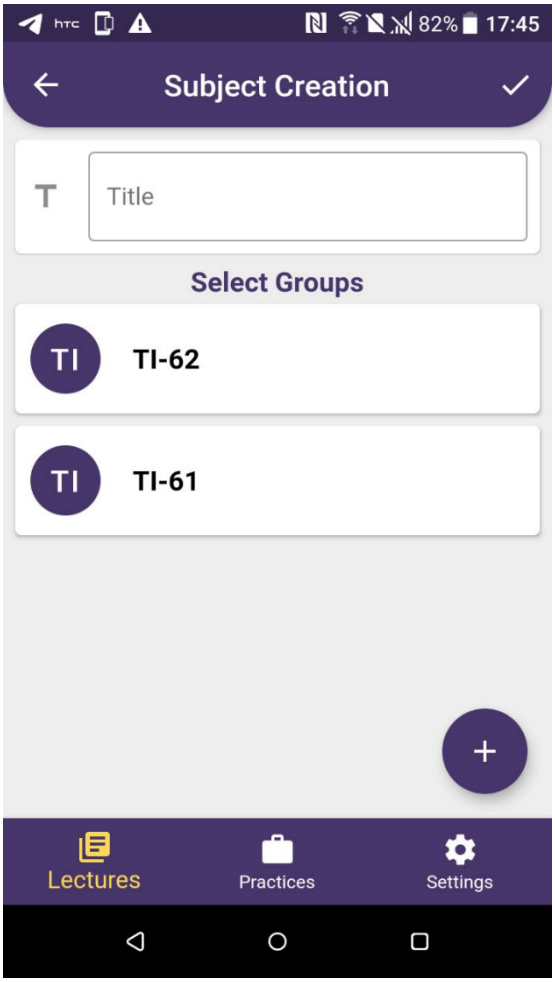

Рисунок 4.11 – Сторінка створення курсу

Проте, для створення курсу потрібно вказати, які групи до нього належать і який тип курсу для кожної з груп. Щоб перейти до сторінки створення групи, потрібно натиснути кнопку зі знаком «+» в нижньому правому кутку інтерфейсу.

На сторінці створення групи, рисунок 4.12, існує два поля обов’язкових для заповнення. Першим полем є назва групи, наступним – список імен студентів відокремлених переходом на новий рядок. Запис всіх імен студентів в одне поле спрощує і пришвидшує створення групи для користувача, адже він може просто скопіювати та вставити цей список.

Після заповнення форми, користувачу потрібно натиснути кнопку підтвердження, що знаходиться в правому кутку верхньої панелі управління.

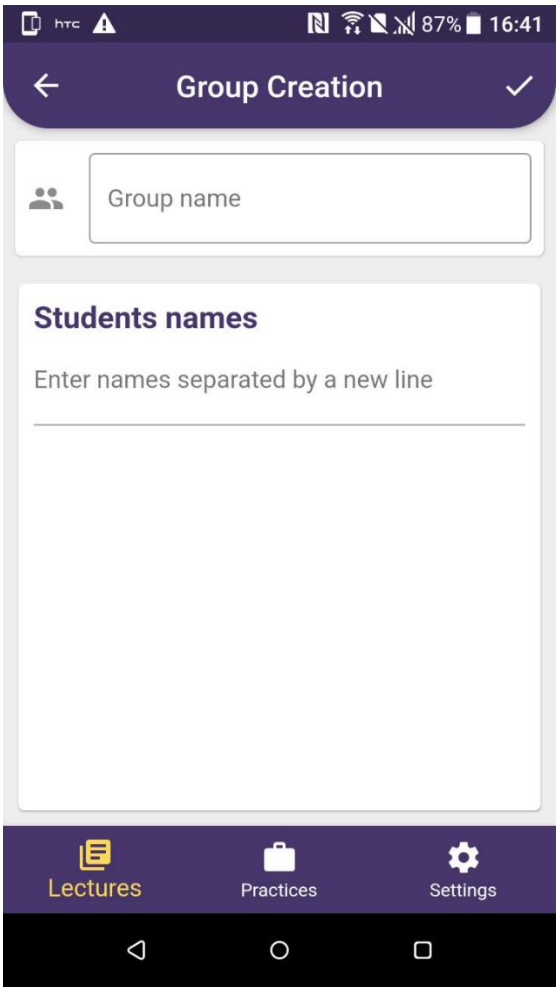

Рисунок 4.12 – Сторінка створення групи

Для того, щоб створити певний курс – потрібно ввести назву курсу та обрати групи, котрі входять до цього курсу натиснувши на елемент інтерфейсу з його назвою і відповідно до кожної групи вказати чи буде цей курс тільки лекційний або тільки практичний або і лекційний, і практичний. За замовчуванням курс для кожної групи є лекційним та практичним. Якщо користувачу потрібну змінити тип курсу, йому потрібно натиснути на ікону розширення на елементі інтерфейсу з назвою відповідної групи і обрати інший тип, рисунок 4.13.

Після заповнення форми, дані проходять валідацію і створюються потрібні курси. Варто зазначити, що лекційний курс створюється один з декількома групами, а практичний курс створюється окремо для кожної групи. Це зроблено, тому що лекційна сесія зазвичай одна для декількох груп водночас. А практичні сесії проводяться окремо для кожної групи.

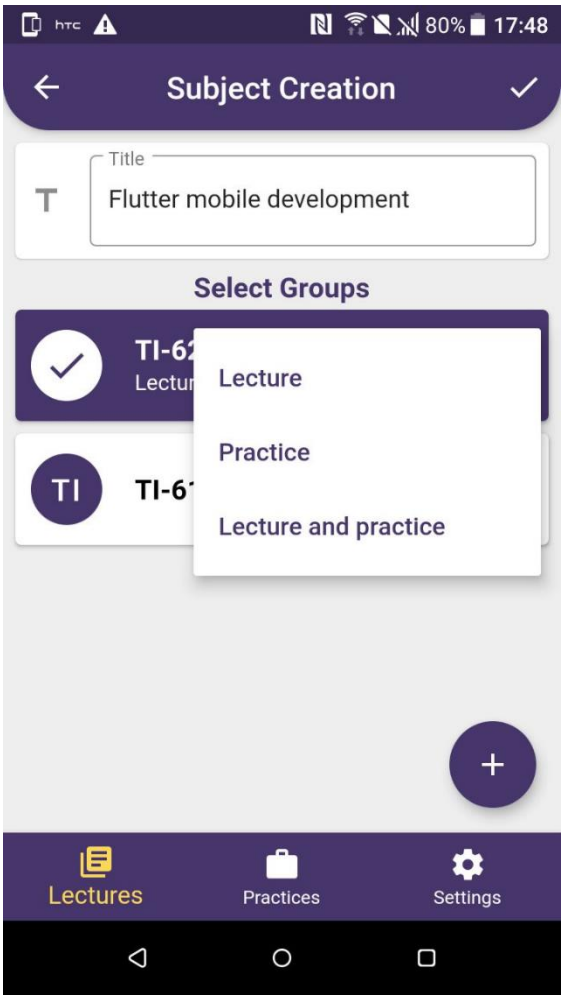

Рисунок 4.13 – Сторінка створення курсу в процесі налаштування

Користувач завжди має змогу видалити створені курси, за допомогою проведенням пальця з правої сторони на ліву сторону на інтерфейсному елементі, який відповідає за існуючий курс, рисунок 4.14. Застосунок зробить запит на підтвердження видалення від користувача, і в разі успіху видалить цільовий курс зі списку.

Якщо ж користувач натисне на інтерфейсний елемент, що пов’язаний з певним курсом, то застосунок прокладе навігаційний шлях до сторінки контролю відвідування студентів лекцій, якщо курс – лекційний, або до сторінки контролю оцінювання практичних робіт студентів, якщо курс – практичний.

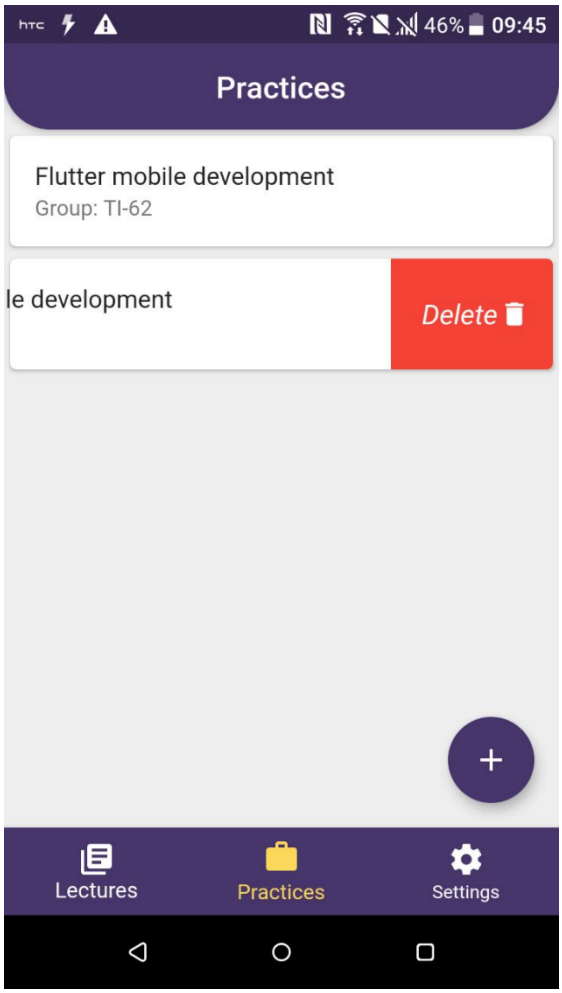

Рисунок 4.14 – Сторінка списку практичних курсів і демонстрація видалення елементу зі списку

На сторінці контролю відвідування та оцінювання студентів розроблена розширена верхня панель управління, яка дає змогу користувачу дізнатись назву курсу, створити першу сесію курсу, рисунок 4.15 та скористатись функціональною можливістю змінити тему вже створеної навчальної сесії або видалити її зі списку.

Коли користувач натискає на кнопку створення навчальної сесії, на екрані

з’являється модальне вікно, що пропонує користувачу дати назву створеній навчальній сесії. Якщо користувач створить навчальну сесію без назви, то вона буде створена з темою за замовчуванням, яку потім можна перейменувати.

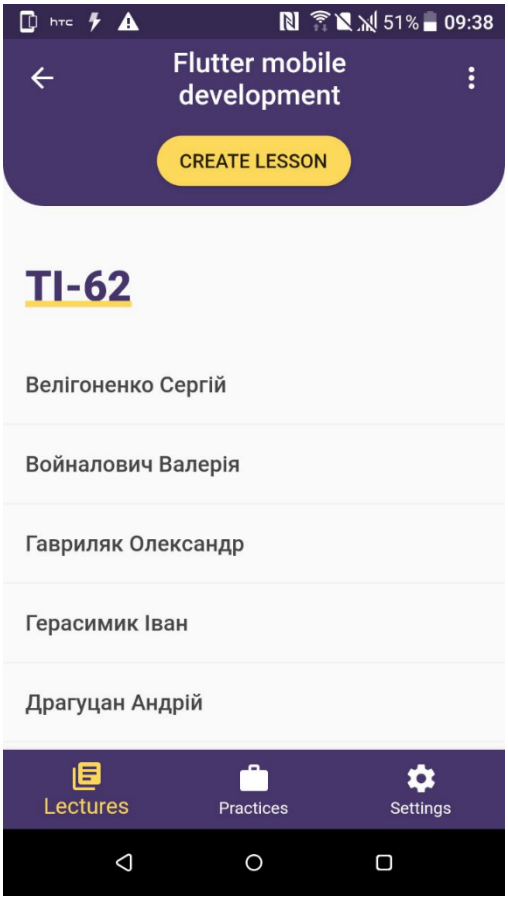

Рисунок 4.15 – Сторінка контролю відвідування студентів без створених навчальних сесій

Після створення навчальної сесії, в лекційному курсі біля імен студентів з’являються прапорці, що дозволяють організувати контроль відвідування учнів, рисунок 4.16. Окрім того, під іменами студентів з’являється додаткова статистична інформація, що інформує користувача про кількість відвідувань окремо кожного студента та відсоток кількості відвідувань від кількості проведених лекцій з даного курсу, рисунок 4.16.

У практичному курсі біля імен студенів з’являються числа, що репрезентують оцінку і дозволяють користувачу виставляти в якості оцінки будь-яке дробове число,

що округлюється до десятих та може бути від'ємним, рисунок 4.17. Оцінки виставляються за допомогою модального вікна, що відкривається після натискання на інтерфейсний елемент з іменем студента. Окрім того, під іменами студентів з'являється додаткова статистична інформація, що інформує користувача про загальну суму усіх отриманих балів студента та його середній бал з даного курсу, рисунок 4.17.

Користувач має змогу переходити між створеними сесіями за допомогою стрілок на верхній панелі управління, рисунок 4.16. Застосунок інформує користувача про порядковий номер поточної навчальної сесії, її тему та дату створення. Якщо наступної сесії в списку не існує, при переході до неї, застосунок пропонує створити нову навчальну сесію.

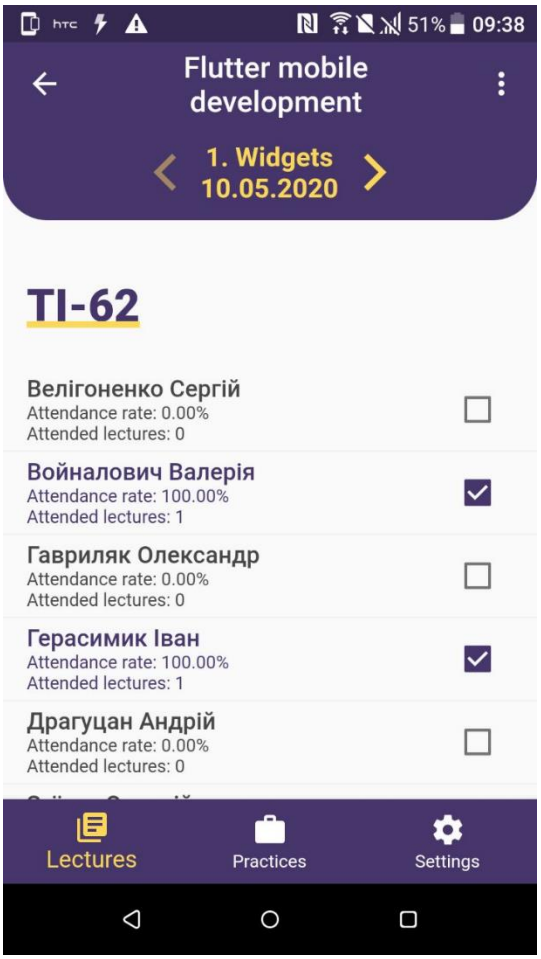

Рисунок 4.16 – Сторінка контролю відвідування студентів

Варто зазначити, що верхня панель управління переходами між сесіями курсу, звертається при прокручуванні списку вниз, щоб звільнити більше місця для іншої інформації. Панель повертається назад при прокручуванні списку вверх.

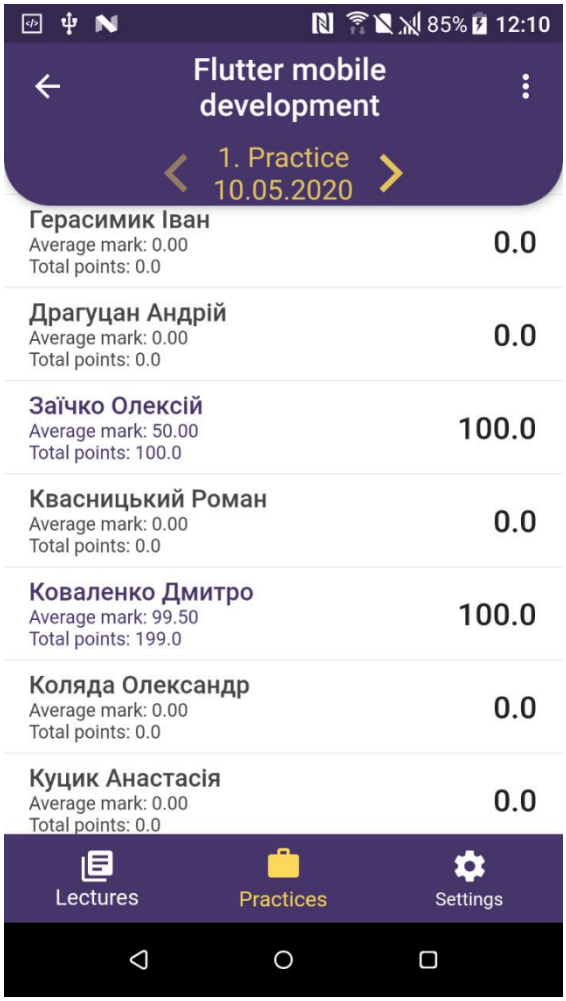

Рисунок 4.17 – Сторінка оцінювання практичних робіт студентів

Користувач має змогу дізнатись детальну інформацію пов’язану з обраним курсом про окремого студента та змінити її користуючись функціональними можливостями сторінки студента, рисунок 4.19.

Щоб змінити дані пов’язані з окремим студентом на обраному курсі, користувач має виконати затискання інтерфейсного елементу пов’язаного з цільовим студентом. Після цього інтерфейсний елемент студент виділяється жовтим кольором, а верхня панель управління навчальним курсом змінюється на панель управління обраним

студентом, рисунок 4.18. Користувач може переглянути сторінку студента, змінити його ім’я або видалити з даного списку.

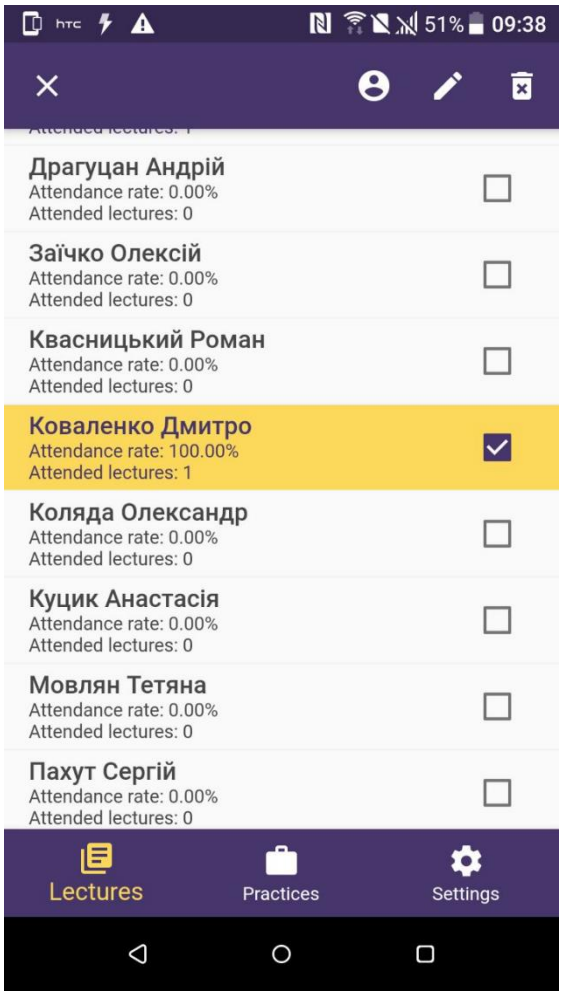

Рисунок 4.18 – Інтерфейс управління даними окремого студента

Натиснувши на знак акаунту в верхній панелі управління, застосунок прокладає навігаційний шлях до сторінки студента, рисунок 4.19. На цій сторінці користувач має змогу контролювати відвідування обраного студента на існуючих лекційних сесіях з цього курсу.

Аналогічно працює сторінка студента пов’язаного з практичним курсом. Проте замість контролю відвідування, користувач має змогу виставляти оцінки за існуючі сесії практичних робіт.

Користувач має змогу відкрити вкладку налаштувань, в якій він може переглядати та керувати інформацією пов’язаною зі створеними ним групами або вийти з акаунту, рисунок 4.20.

Аналогічно до того, як користувач має змогу змінити ім’я студента або видалити його зі списку групи в межах певного курсу, так само користувач має змогу виконати дані дії в масштабі всього застосунку. Це дозволяє створювати нові курси з оновленою інформацією про групи. Також, є можливість видалити групу зі списку груп, проте створені курси з цією групою залишаться і інформація в них також буде збережена. Видаляючи групу, користувач видаляє шаблон даної групи, потрібний для створення курсів.

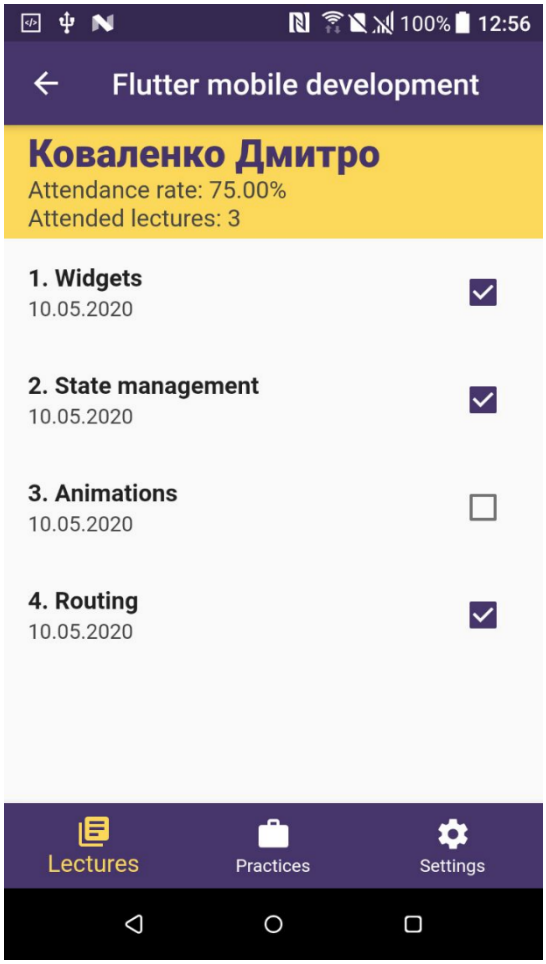

Рисунок 4.19 - Сторінка контролю організації відвідувань певного студента

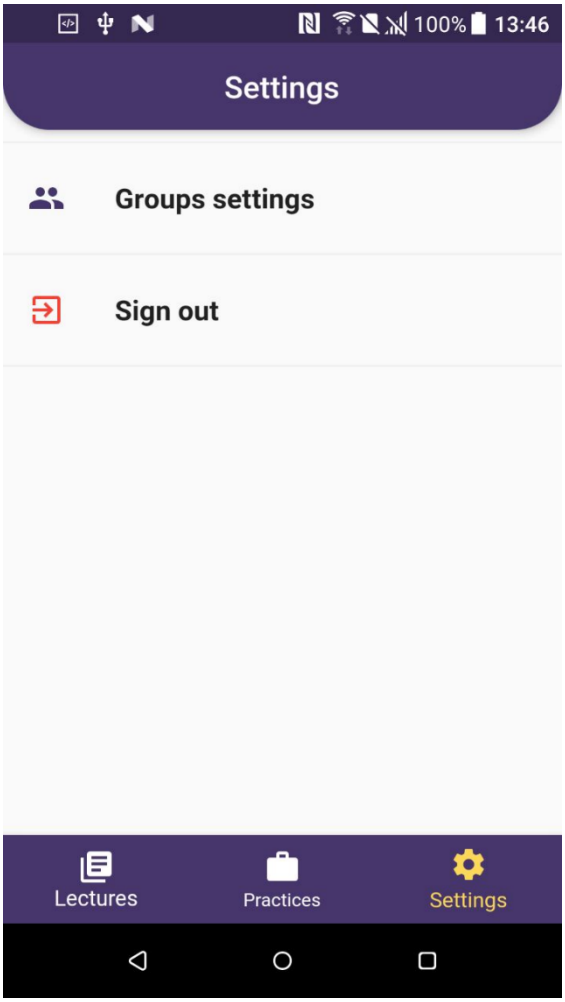

Рисунок 4.20 – Сторінка налаштувань

4.4 Висновки до розділу

У розділі було розкрито структуру програмного продукту, методи, що були використані при розробці та керівництво користувача, де були описані всі функціональні можливості застосунку, якими явно керує користувач.

Кожна представлена сторінка несе в собі важливі та необхідні функціональні можливості. Застосунок створений з інтуїтивним інтерфейсом та зручним досвідом користувача. Основні функціональні можливості представлені максимально просто, в той час як другорядні – приховані від ока користувача, щоб уникнути перевантаження інтерфейсу. Щоб використати другорядні функціональні можливості

користувач має вказати цільові елементи інтерфейсу. Використання таких методів, успішно полегшує користування мобільним застосунком.

Керівництво користувача містить необхідні інструкції для швидкого ознайомлення із усіма функціональними можливостями системи.

ВИСНОВКИ

Було визначено та проаналізовано проблему організації контролю відвідування та оцінювання студентів.

В ході виконання розглянутої роботи було розроблено програмне забезпечення у вигляді мобільного застосунку для організації контролю відвідування та оцінювання студентів.

Досліджені сучасні цифрові методи оцінювання та організації контролю відвідування та оцінювання студентів. Проведений порівняльний аналіз існуючих аналогів, в якості мобільних застосунків, і на основі поставлених цілей та виявлених недоліків, знайдених в аналогах, визначені функціональлні можливості інструментарію.

Створена концепт-діаграма користувацького досвіду (UX) та визначений дизайн користувацького інтерфейсу. Спроектована модель класів. Обрані технології Flutter та Firebase для реалізнації функціональних можливостей застосунку.

Була налаштована система Firebase, в якості back-end частини, і потім інтегрована до мобільного застосунку для отримання доступу до хмарного сховища. Розроблені всі потрібні частини застосунку та об’єднані в фінальний програмний продукт з спроектованим дизайном.

Мобільний застосунок протестований на наявність помилок в програмному коді та покращений, за допомогою їх усунення.

Програмний продукт спроектовано та розроблено з урахуванням майбутнього розширення функціональних можливостей.

Виконано усі поставлені задачі, мета роботи досягнута, а програмний продукт готовий до масового застосування користувачами в предметній області.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 2017 Retrospective: A Monumental Year for the App Economy [Електронний ресурс]/Режим доступу:https://www.appannie.com/en/insights/market-data/app-annie-2017-retrospective/
2. 2018 — рік мобільних додатків в МФО [Електронний ресурс]/Режим доступу:https://minfin.com.ua/2018/01/30/32121301/
3. Beginning App Development with Flutter / Rap Payne, 2019. – 334 с.
4. Flutter - Beautiful native apps in record time [Електронний ресурс]/Режим доступу: https://flutter.dev/
5. Dart programming language [Електронний ресурс]/Режим доступу: https://dart.dev/
6. Firebase [Електронний ресурс]/Режим доступу: https://firebase.google.com/docs/
7. Cloud Firestore [Електронний ресурс]/Режим доступу: https://cloud.google.com/firestore
8. Beginning Flutter: A Hands On Guide to App Development / Marco L. Napoli, 2019. – 528 с.
9. Flutter in Action / Eric Windmill, 2019. – 368 с.
10. The Dart Programming Language / Gilad Bracha, 2015. – 224 с.

ДОДАТОК А

Інструментальні засоби мобільного застосунку
для організації контролю
відвідувань студентів

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ62287_20А 1

Аркушів 2

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6287_20Б 2	Записка Коваленко Д.Р. ТІ-62.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6287_20Б 12-1	registly.dart	Компонент вкладеної навігації
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6287_20Б 12-2	model.dart	Компонент керування станом програми
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6287_20Б 12-3	Firebase_service.dart	Компонент інтеграції з бек-енд частиною застосунку

ДОДАТОК Б

Інструментальні засоби мобільного застосунку
для організації контролю
відвідувань студентів

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ62287_20Б 2

Аркушів 18

Київ – 2020

registly.dart

```
// enum of tabs
enum TabItem { lecture, practice, settings }

//class of stacked navigation
class Registly extends StatefulWidget {
  @override
  _RegistlyState createState() => _RegistlyState();

  @override
  StatefulWidget get widget => null;
}

//state class of stacked navigation
class _RegistlyState extends State<Registly> {
  TabItem currentTab = TabItem.lecture;
  TabController tabController;

  Map<TabItem, GlobalKey<NavigatorState>> navigatorKeys = {
    TabItem.lecture: GlobalKey<NavigatorState>(),
    TabItem.practice: GlobalKey<NavigatorState>(),
    TabItem.settings: GlobalKey<NavigatorState>(),
  };

  @override
  void initState() {
    super.initState();
  }

  void _selectTab(TabItem tabItem) {
    setState(() {
      currentTab = tabItem;
    });
  }
```

```
@override
Widget build(BuildContext context) {
  return WillPopScope(
    onWillPop: () async =>
        await navigatorKeys[currentTab].currentState.maybePop(),
    child: Consumer<Model>(
      builder: (context, model, child) {
        if (model.firebase != null && model.isSignedIn) {
          return Scaffold(
            body: Stack(
              children: <Widget>[
                _buildTabNavigator(TabItem.lecture),
                _buildTabNavigator(TabItem.practice),
                _buildTabNavigator(TabItem.settings),
              ],
            ),
            bottomNavigationBar: BottomNavigation(
              currentTab: currentTab,
              onSelectTab: _selectTab,
            ),
          );
        } else if (model.showLoader) {
          return Scaffold(
            body: Center(
              child: SpinKitFadingCube(size: 75.0, color: AppColors.appBar),
            ),
          );
        } else {
          model.currentUserSignIn();
          return MailLogin();
        }
      },
    ),
```

```
    );
  }

  Widget _buildTabNavigator(TabItem tabItem) {
    return Offstage(
      offstage: currentTab != tabItem,
      child: TabNavigator(
        navigatorKey: navigatorKeys[TabItem],
        tabItem: tabItem,
      ),
    );
  }
}
```

model.dart

```
class Model with ChangeNotifier {
  FirebaseService firebase;

  List<Group> groups;
  List<Lecture> lectures;
  List<Practice> practices;

  bool _showLoader = false;
  bool _isSignedIn = false;

  bool get isSignedIn => _isSignedIn;

  set isSignedIn(bool isSignedIn) {
    _isSignedIn = isSignedIn;
    notifyListeners();
  }

  bool get showLoader => _showLoader;
```

```
set showLoader(bool showLoader) {
  _showLoader = showLoader;
  notifyListeners();
}

// will sign firebase in if user already signed in
void currentUserSignIn() async {
  firebase = await FirebaseService.init();
  if (firebase != null) {
    showLoader = true;
    loginUser().then((_) {
      isSignedIn = true;
      showLoader = false;
    });
  }
}

void signInAccount() async {
  if (firebase == null) {
    showLoader = true;
    firebase = await FirebaseService.signInWithGoogle();
    loginUser().then((_) {
      isSignedIn = true;
      showLoader = false;
    });
  }
}

Future<bool> signInEmailAccount({String email, String password}) async {
  firebase = await FirebaseService.signInWithEmail(email, password);

  if (firebase != null) {
    loginUser().then((_) {
      isSignedIn = true;
```

```
    });
    return true;
  } else
    return false;
}

Future<bool> signUpEmailAccount({String email, String password}) async {
  if (firebase == null) {
    firebase = await FirebaseService.signUpWithEmail(email, password);
    print(firebase);
    if (firebase != null) {
      return true;
    } else {
      return false;
    }
  }
  return false;
}

Future<bool> recoverPassword({@required String email}) async {
  return await FirebaseService.recoverPassword(email);
}

Future loginUser() async {
  firebase.updateUserData(firebase.firebaseUser);
  await downloadGroups();
  await downloadLectures();
  await downloadPractices();
  notifyListeners();
}

void signOutGoogle() {
  FirebaseService.signOutGoogle();
  lectures = [];
```

```
  practices = [];
  groups = [];
  firebase = null;
  showLoader = false;
  isSignedIn = false;
  notifyListeners();
}

//////////////////////
/* Download functions */
//////////////////////
Future downloadGroups() async {
  var result = await firebase.getAllGroups();
  groups = result.documents.map((doc) => Group.fromJSON(doc.data)).toList();
  if (groups == null) groups = [];
  notifyListeners();
}

Future downloadLectures() async {
  var result = await firebase.getAllLectures();
  lectures =
      result.documents.map((doc) => Lecture.fromJSON(doc.data)).toList();
  if (lectures == null) lectures = [];
  notifyListeners();
}

Future downloadPractices() async {
  var result = await firebase.getAllPractices();
  practices =
      result.documents.map((doc) => Practice.fromJSON(doc.data)).toList();
  if (practices == null) practices = [];
  notifyListeners();
}
```

```
//////////////////////
/* Creation functions */
//////////////////////
void addGroup(String title, List<Student> students) {
  Group group = Group.byTitle(title, students);
  groups.add(group);
  uploadGroup(group);
  notifyListeners();
}

void addSubjects(String title, List<GroupLesson> groupLessons) {
  groupLessons
      .where((lesson) =>
          lesson.lessonType == LessonType.practice ||
          lesson.lessonType == LessonType.lectureAndPractice)
      .forEach((lesson) {
    Practice practice = Practice.byTitle(title, lesson.group);
    this.practices.add(practice);
    uploadPractice(practice);
  });

  List<Group> groups = groupLessons
      .where((lesson) =>
          lesson.lessonType == LessonType.lecture ||
          lesson.lessonType == LessonType.lectureAndPractice)
      .map((lesson) => lesson.group)
      .toList();

  if (groups.isNotEmpty) {
    Lecture lecture = Lecture.byTitle(title, groups);
    this.lectures.add(lecture);
    uploadLecture(lecture);
  }
```

```
  notifyListeners();
}

//////////////////////
/* Setter functions */
//////////////////////
void setMark(
    {@required String practiceId,
    @required int studentIndex,
    @required int markIndex,
    @required double mark}) {
  practices
      .firstWhere((practice) => practice.id == practiceId)
      .group
      .studentList[studentIndex]
      .marksList[markIndex] = mark;
}

//////////////////////
/* Upload functions */
//////////////////////
void uploadGroup(Group group) => firebase.addGroup(group.toJSON());

void uploadPractice(Practice practice) =>
    firebase.addPractice(practice.toJSON());

void uploadLecture(Lecture lecture) => firebase.addLecture(lecture.toJSON());

//////////////////////
/* Update functions */
//////////////////////
void updateLecture(String id) => firebase.updateLecture(
    lectures.firstWhere((lecture) => lecture.id == id).toJSON(), id);
```

```
void updatePractice(String id) => firebase.updatePractice(
    practices.firstWhere((practice) => practice.id == id).toJSON(), id);

void updateGroup(String id) => firebase.updateGroup(
    groups.firstWhere((group) => group.id == id).toJSON(), id);

////////////////////////////////////
/* Edit functions */
////////////////////////////////////

void editLectureTitle(int index, int lectureIndex, String newTitle) {
  lectures[index].lectureTitles[lectureIndex] = newTitle;
  updateLecture(lectures[index].id);
  notifyListeners();
}

void editPracticeTitle(int index, int practiceIndex, String newTitle) {
  practices[index].practiceTitles[practiceIndex] = newTitle;
  updatePractice(practices[index].id);
}

////////////////////////////////////
/* List student action functions */
////////////////////////////////////
void checkStudentAttendance(
      int index, int groupIndex, int studentIndex, int lectureIndex) =>
    this
          .lectures[index]
          .groups[groupIndex]
          .studentList[studentIndex]
          .attendenceList[lectureIndex] =
      !this
          .lectures[index]
          .groups[groupIndex]
```

```
            .studentList[studentIndex]
            .attendenceList[lectureIndex];

void editLectureStudentName(
        int index, int groupIndex, int studentIndex, String name) =>
    this
        .lectures[index]
        .groups[groupIndex]
        .studentList[studentIndex]
        .fullName = name;

void removeLectureStudent(int index, int groupIndex, int studentIndex) => this
    .lectures[index]
    .groups[groupIndex]
    .studentList
    .removeAt(studentIndex);

// void editPracticeStudentName(int index, int studentIndex, String name) =>
//     this.practices[index].group.studentList[studentIndex].fullName = name;

void editPracticeStudentName(String id, String studentId, String name) =>
    practices.where((practice) => practice.group.id == id).forEach(
        (practice) => practice.group.studentList
            .firstWhere((student) => student.id == studentId)
            .fullName = name);

// void removePracticeStudent(int index, int studentIndex) =>
//     this.practices[index].group.studentList.removeAt(studentIndex);

void removePracticeStudent(String id, String studentId) => practices
    .where((practice) => practice.group.id == id)
    .forEach((practice) => practice.group.studentList
        .removeWhere((student) => student.id == studentId));
```

```
void editGroupStudentName(String id, String studentId, String name) {
  groups
      .firstWhere((group) => group.id == id)
      .studentList
      .firstWhere((student) => student.id == studentId)
      .fullName = name;

  updateGroup(id);
  //editPracticeStudentName(id, studentId, name);
}

void removeGroupStudent(String id, String studentId) {
  groups
      .firstWhere((group) => group.id == id)
      .studentList
      .removeWhere((student) => student.id == studentId);
  updateGroup(id);
}

//////////////////////
/* Remove functions */
//////////////////////
void removeLectureSubject(int index) {
  String id = lectures[index].id;
  lectures.removeAt(index);
  firebase.deleteLecture(id);
  notifyListeners();
}

void removePracticeSubject(int index) {
  String id = practices[index].id;
  practices.removeAt(index);
  firebase.deletePractice(id);
  notifyListeners();
```

```
  }

  void removeGroup(int index) {
    String id = groups[index].id;
    groups.removeAt(index);
    firebase.deleteGroup(id);
    notifyListeners();
  }
}
```

firebase_service.dart

```
class FirebaseService {
  static FirebaseAuth _auth = FirebaseAuth.instance;
  static GoogleSignIn _googleSignIn = GoogleSignIn();

  FirebaseUser firebaseUser;

  static final _db = Firestore.instance;

  Map<String, CollectionReference> _collectionReference;

  DocumentReference _documentReference;

  FirebaseService(FirebaseUser user) {
    this.firebaseUser = user;
  }

  static Future<FirebaseService> init() async {
    final currentUser = await FirebaseAuth.instance.currentUser();
    if (currentUser != null)
      return FirebaseService(currentUser);
    else
      return null;
```

```
}

static Future<FirebaseService> signInWithGoogle() async {
  GoogleSignInAccount googleUser;
  if (await _googleSignIn.isSignedIn())
    googleUser = await _googleSignIn.signInSilently();
  else
    googleUser = await _googleSignIn.signIn();

  final GoogleSignInAuthentication googleAuth =
      await googleUser.authentication;
  AuthCredential credential = GoogleAuthProvider.getCredential(
      accessToken: googleAuth.accessToken, idToken: googleAuth.idToken);

  final FirebaseUser user =
      (await _auth.signInWithCredential(credential)).user;

  final FirebaseUser currentUser = await _auth.currentUser();

  return FirebaseService(user);
}

static Future<FirebaseService> signInWithEmail(
    String email, String password) async {
  AuthResult result;
  try {
    result = await _auth.signInWithEmailAndPassword(
        email: email, password: password);
  } on PlatformException catch (error) {
    print(error.code.toString());
  } finally {
    if (result != null) {
      FirebaseUser user = result.user;
```

```
      if (user != null) {
        if (!user.isEmailVerified) return null;
        return FirebaseService(user);
      } else
        return null;
    }
  }
}

static Future<FirebaseService> signUpWithEmail(
    String email, String password) async {
  AuthResult result;
  try {
    result = await _auth.createUserWithEmailAndPassword(
        email: email, password: password);
  } on PlatformException catch (error) {
    print(error.code.toString());
  } finally {
    if (result != null) {
      FirebaseUser user = result.user;

      if (user != null) {
        await user.sendEmailVerification();
        return FirebaseService(user);
      } else
        return null;
    }
  }
}

static Future<bool> recoverPassword(String email) async {
  bool isSent = true;
  try {
    await _auth.sendPasswordResetEmail(email: email);
```

```
  } on PlatformException catch (error) {
    print(error.code.toString());
    isSent = false;
  } finally {
    return isSent;
  }
}

static signOutGoogle() async {
  _auth.signOut();
  _googleSignIn.signOut();
}

void updateUserData(FirebaseUser user) async {
  _documentReference = _db.collection('users').document(user.uid);
  _collectionReference = {
    'groups': _documentReference.collection('groups'),
    'practices': _documentReference.collection('practices'),
    'lectures': _documentReference.collection('lectures'),
  };

  return _documentReference.setData({
    'uid': user.uid,
    'email': user.email,
    'photoURL': user.photoUrl,
    'displayName': user.displayName,
    'lastSeen': DateTime.now()
  }, merge: true);
}

Future<DocumentReference> addGroup(Map data) =>
    _collectionReference['groups'].add(data);

Future<DocumentReference> addPractice(Map data) =>
```

```
    _collectionReference['practices'].add(data);

Future<DocumentReference> addLecture(Map data) =>
    _collectionReference['lectures'].add(data);

Future<void> updateLecture(Map data, String id) =>
    _collectionReference['lectures']
        .where('id', isEqualTo: id)
        .getDocuments()
        .then((snapshot) => snapshot.documents
            .forEach((doc) => doc.reference.updateData(data)));

Future<void> updatePractice(Map data, String id) =>
    _collectionReference['practices']
        .where('id', isEqualTo: id)
        .getDocuments()
        .then((snapshot) => snapshot.documents
            .forEach((doc) => doc.reference.updateData(data)));

Future<void> updateGroup(Map data, String id) =>
    _collectionReference['groups']
        .where('id', isEqualTo: id)
        .getDocuments()
        .then((snapshot) => snapshot.documents
            .forEach((doc) => doc.reference.updateData(data)));

Future deleteLecture(String id) => _collectionReference['lectures']
    .where('id', isEqualTo: id)
    .getDocuments()
    .then((snapshot) =>
        snapshot.documents.forEach((doc) => doc.reference.delete()));

Future deletePractice(String id) => _collectionReference['practices']
    .where('id', isEqualTo: id)
```

```
      .getDocuments()
      .then((snapshot) =>
          snapshot.documents.forEach((doc) => doc.reference.delete()));

  Future deleteGroup(String id) => _collectionReference['groups']
      .where('id', isEqualTo: id)
      .getDocuments()
      .then((snapshot) =>
          snapshot.documents.forEach((doc) => doc.reference.delete()));

  Future<QuerySnapshot> getAllGroups() =>
      _collectionReference['groups'].getDocuments();

  Future<QuerySnapshot> getAllPractices() =>
      _collectionReference['practices'].getDocuments();

  Future<QuerySnapshot> getAllLectures() =>
      _collectionReference['lectures'].getDocuments();
}
```

ДОДАТОК В

Інструментальні засоби мобільного застосунку
для організації контролю
відвідувань студентів

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ62287_20В 3

Аркушів 11

Київ – 2020

АНОТАЦІЯ

Додаток містить опис основних програмних класів інструментальних засобів мобільного застосунку для організації контролю відвідувань студентів, що виконують деякі головні функціональні можливості програми, а саме:

— Можливість вкладеної навігації, що дозваляє мати окрему навігаційну історію шляхів для кожної вкладки.

— Можливість керувати всіма даними мобільного застосунку.

— Можливість комунікувати з back-end частиною застосунку, що базується на сервісі Firebase.

Мобільний застосунок розроблений за допомогою мови програмування Dart з використанням інструментальних засобів для створення UI – Flutter та сервісу Firebase.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ 86
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 87
3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ 89
4. ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУЮТЬСЯ 90
5. ВИКЛИК І ЗАВАНТАЖЕННЯ 91
6. ВХІДНІ ДАНІ 92
7. ВИХІДНІ ДАНІ 93

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис основних програмних класів інструментальних засобів мобільного застосунку для організації контролю відвідувань студентів, що виконують деякі головні функціональні можливості програми.

У додатку Б міститься програмний код основних програмних модулів.

Мобільний застосунок для роботи потребує мобільний пристрій з операційною системою Android версії 4.0.0 і вище або пристрій з операційною системою iOS версії 8.0 і вище.

Мобільний застосунок розроблений за допомогою мови програмування Dart з використанням інструментальних засобів для створення UI – Flutter та сервісу Firebase.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Інструментальні засоби мобільного застосунку для організації контролю відвідувань студентів, надають такі функціональні можливості програми:

1. Наявна можливість атворизації та регистрації.
2. Наявна можливість авторизації за допомогю акаунту Google.
3. Наявна можливість відновлення паролю до існуючого акаунту.
4. Наявна можливість входу до системи без авторизації та підключення до інтернету, якщо користувач вже входив до свого акаунту і не виходив з нього.
5. Наявна можливість автоматичного збереження всіх даних до хмарного сховища. Якщо пристрій не має підключення до інтернету, дані мають записуватись на локальне сховище і завантажуватись після відновлення підключення.
6. Наявна можливість створення груп зі студентами, предметів з класифікацією лекції та практики.
7. Наявна можливість зміни назви групи та імен окремих студентів, а також видалення студентів зі списку групи.
8. Наявна можливість видалення предметних практик або лекцій.
9. Наявна можливість автоматичної назви та нумерації уроків лекцій та практик кожного предмета.
10. Наявна можливість зміни назви або видалення уроку лекції або практики.
11. Наявна можливість автоматичної прив’язки дати до уроку при його створенні.
12. Наявна можливість зміни імені або видалення студента в межах одного предмету.
13. Наявна можливість переглянути сторінку окремого студента на котрій зібрані дані, щодо його відвідування та отриманих оцінок, а також відсоток

відвідувань та їх кількість на лекціях або середнє арифметичне оцінок та їх сума на практиках.

14. Наявна можливість на сторінці студента змінювати дані, щодо його відвідувань та отримані оцінки.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, додаток складається з трьох головних модулів:

— шар віджетів.

— шар моделей.

— шар зв’язку з back-end частиною застосунку, сервісом Firebase.

Шар віджетів містить велику кількість класів, що слугують для демонстрації користувацького інтерфейсу та реактивних змін на екрані за допомогую Stateless та Stateful віджетів.

Шар моделей містить в собі класи, що використовуються для моделювання сутностей, якими оперує мобільний застосунок.

Шар зв’язку зв’язку з back-end частиною застосунку, сервісом Firebase містить клас з методами для авторизації та реєстрації різними методами, а також для створення, зчитування, видалення та оновлення даних в Cloud Firestore.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання мобільного застосунку користувач повинен мати мобільний пристрій з операційною системою Android версії 4.0.0 і вище або пристрій з операційною системою iOS версії 8.0 і вище.

Для першого використання застосунку користувач має мати підключення до мережі інтернет для авторизації у систему інструментальних засобів. Після цього, авторизація є автоматичною і всі зміни в системі зберігаються локально, в разі відсутності підключення до мережі інтернет. Після відновлення підключення до мережі всі дані синхронізуються.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програма потребує інсталяції і її інсталяційний файл для операційної системи Android можна завантажити за посиланням https://drive.google.com/open?id=17KQxZ83jaEkBP2WiO_CQoK1E_q_Fz6I8.

Щоб інсталювати програму, достатньо запустити файл на мобільному пристрої, з оперіційною системою Android. Після інсталяції, програма готова до використання.

ВХІДНІ ДАНІ

Вхідна інформація для інструментальних засобів мобільного застосунку:

1. Вхідними даними системи аутентифікації мають бути електронна пошта та пароль або існуючий акаунт Google.
2. Вхідними даними системи створення груп та предметів мають бути імена студентів, назви груп, назви предметів та їх класифікація.
3. Вхідними даними системи контролю відвідувань та оцінювання студентів на лекціях і практиках мають бути назви та дати створення уроку, мітка присутності на лекції та оцінка на практиці.

ВИХІДНІ ДАНІ

Вихідна інформація для інструментальних засобів мобільного застосунку:

1. Вихідними даними системи аутентифікації мають бути інформація привязана до даного акаунту та дані для подальшої автоматичної аутентифікації.
2. Вихідними даними системи створення груп та предметів мають бути сутності студентів, груп та предметів.
3. Вихідними даними системи контролю відвідувань та оцінювання студентів на лекціях і практиках мають бути збережені вхідні дані, відсоток відвідувань та їх кількість на лекціях або середнє арифметичне оцінок та їх сума на практиках для кожного студента.

ДОДАТОК Г

Інструментальні засоби мобільного застосунку
для організації контролю
відвідувань студентів

Публікації

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ62287_20Г 4

Аркушів 2

Київ – 2020

Дмитро Коваленко
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ
koditka@gmail.com
Науковий керівник – С. В. Титенко, О. О. Гагарін

ЕЛЕКТРОННИЙ ЖУРНАЛ УСПІШНОСТІ СТУДЕНТІВ НА БАЗІ МОБІЛЬНОГО ЗАСТОСУНКУ

На сьогодні викладачі в університетах ще досі використовують старі методи для контролю відвідувань та оцінювання студентів. Великий обсяг паперів зі списками студентів для різних груп і предметів не є достатньо надійним та зручним способом організації інформації про навчальний процес.

Серед тенденцій сучасного світу усе більших масштабів набирає використання мобільних застосунків для спрощення доступу до необхідної інформації. Тому вбачається доцільним розробка мобільного застосунку, який вирішує дану проблему і є легким у використанні.

Для вирішення даної проблеми були створені такі аналоги: “Class Register”, “Attendance register”, “TCMS”. Проте, проблема залишається не вирішеною, адже аналоги мають незручний інтерфейс і складний досвід користувача (UX) або недостатні функціональні можливості.

Пропонуються концепт-діаграми мобільного застосунку, що демонструють користувацький досвід мобільного застосунку. На рисунку 1 показано, як викладач в два дотики може перейти до списків студентів потрібної йому групи на потрібній парі, як на лекції так і практиці. З екрану списку групи нової лекції або практики користувач може переходити на сторінки з інформацією про попередньо проведені пари та змінювати старі та тільки-но створені дані. Щоб відмітити присутність студента на лекції або виставити оцінку потрібен один дотик. Окрім того, біля імені студента завжди відображена інформація про його середньостатистичну присутність на лекціях та його середній бал.

Рис. 1. Концепт-діаграма користувацького досвіду

Рисунок 2 демонструє, як швидко викладач може дізнатися всю статистку про обраного студента, натиснувши на його ім’я в одному з вище названих списків. Користувач отримає відомості про присутність студента на кожній лекції та при бажанні має можливість виставити оцінки за проведені практики. Додатково буде виведена інформація про середньостатистичні відвідування та оцінки.

На рисунку 3 показано як легко і швидко можна створити предмет і автоматично створити лекції з цього предмету, та якщо необхідно практики. Якщо ж група ще не записана, то її можна легко додати, вказавши її назву та вставивши список імен студентів відокремлених новим рядком.

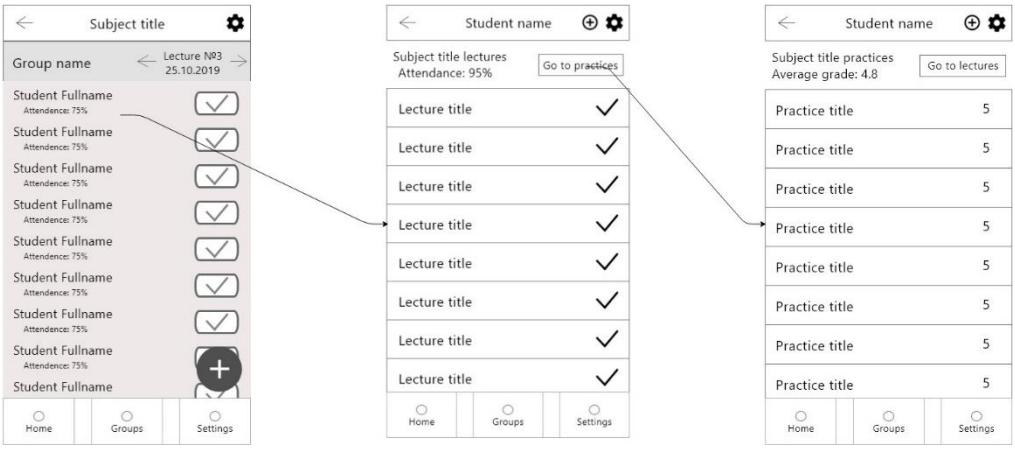

Рис. 2. Концепт-діаграма отримання детальної інформації про студента

Рис. 3. Концепт-діаграма створення нового предмету та нової групи

Для розробки мобільного застосунку вбачається за доцільне використання фреймворку Flutter на мові програмування Dart від компанії Google. Flutter – це новий інструментарій від Google для створення мобільних застосунків. Він дає розробникам змогу швидко будувати виразний інтерфейс.

Список використаних джерел

1. Google. (n.d.). Flutter – beautiful native apps in record time [Електронний ресурс]. Отримано 31 жовтня 2019. Режим доступу: https://flutter.dev/

Анотація. Коваленко Д. Мобільні способи організації та контролю відвідування студентів. *У статті проаналізована проблема незручного методу роботи зі списками студентів і представлене її вирішення у вигляді мобільного застосунку. Наведені існуючі аналоги, що не вирішують проблему до кінця. Подані детально описані концепт-діаграми користувацького досвіду.*

Ключові слова: *мобільні додатки, концепт-діаграми, користувацький досвід, методи організації та контролю відвідування студентів.*

Abstract. Kovalenko D. Mobile ways to organize and control student attendance. *The article analyzes the problem of inconvenient method of working with student lists and presents its solution in the form of a mobile application. Existing analogues that do not solve the problem to the end are given. The user experience concept diagrams are presented in detail.*

Keywords: *mobile applications, concept diagrams, user experience, methods of organizing and controlling student attendance.*

Аннотация. Коваленко Д. Мобильные способы организации и контроля посещения студентов. *В статье проанализирована проблема неудобного метода работы со списками студентов и представлено ее решения в виде мобильного приложения. Приведенные существующие аналоги, не решают проблему до конца. Представленные подробно описаны концепт-диаграммы пользовательского опыта.*

Ключевые слова: *мобильные приложения, концепт-диаграммы, пользовательский опыт, методы организации и контроля посещения студентов.*